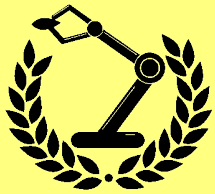
Escuela
Politécnica
Superior

Gestión energética basada en protocolos IoT y procesamiento de datos en la nube



Grado en Ingeniería Robótica

Trabajo Fin de Grado

Autor:
Víctor Germán Merino Kosina
Tutor/es:
Francisco Javier Ferrández Pastor

Escuela
Politécnica
Superior

Julio 2019

Gestión energética basada en protocolos IoT y procesamiento de datos en la nube

Autor
Víctor Germán Merino Kosina

Tutor/es
Francisco Javier Ferrández Pastor
Departamento de Tecnología Informática y Computación

Grado en Ingeniería Robótica

Escuela Politécnica Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Julio 2019

Motivación

Varias razones han motivado a la realización de este trabajo, entre las que se encuentran:

1. **Aprender sobre el Internet de las Cosas y la Industria 4.0.** En este proyecto he conseguido aumentar mis conocimientos en un sector que está en auge hoy en día y que se prevé tenga mucho futuro.
2. **Continuar mi formación en el uso de herramientas para la gestión y la eficiencia energéticas.** Al realizar las prácticas en una empresa de este sector vi una oportunidad de aprovechar esos conocimientos adquiridos y progresar aún más sobre un ámbito de mi interés.
3. **Conseguir realizar un prototipo funcional en el ámbito doméstico por mi mismo, con apoyos puntuales de mi tutor.** Tener la posibilidad de hacer un proyecto real y funcional por mi cuenta me ha motivado en gran medida.
4. **Aportar mi granito de arena en mejorar la eficiencia energética global.** Aunque se consigan unos resultados interesantes a nivel académico, este trabajo puede servir de base para proyectos más avanzados sobre la gestión energética en la nube que pueden generar oportunidades de negocio.

Agradecimientos

Me gustaría empezar por dar las gracias a mi tutor por aceptar mi propuesta de realizar mi Trabajo Fin de Grado con él sobre el Internet de las Cosas y la gestión energética, especialmente, porque es un tema muy innovador.

Aprovecho para dar las gracias a los compañeros de clase por habernos ayudado mutuamente a superar estos cuatro años del Grado y a nuestros profesores por haber dedicado su esfuerzo y sus conocimientos a la primera promoción de Ingeniería Robótica en España.

Quiero agradecer a mis amigos por estar ahí siempre aconsejándome en las decisiones por tomar y apoyándome en las ya tomadas. Sin vosotros este trabajo, el Grado y mi vida sería mucho más difícil.

Por último, y no menos importante, agradezco a mi familia el apoyo incondicional al elegir este Grado en Ingeniería Robótica y en ayudarme en todos los aspectos de la vida para seguir logrando objetivos, tanto personales como académicos y profesionales.

Estas personas forman una parte importante de mi vida y es a ellas a quienes dedico este trabajo.

Dale pescado a un hombre
hambriento y comerá un día. Enséñale
a pescar y comerá toda su vida.

Lao-Tse.

Las cosas no se cambian
para que sean diferentes, sino
para que sean mejores.

Elon Musk.

Índice general

Índice de figuras

Índice de Códigos

1 Introducción

Estamos viviendo una época en la que la eficiencia y la sostenibilidad están a la orden del día. Tenemos a la mayoría de los gobiernos de todo el mundo preocupados por el calentamiento global. Las personas y las empresas cada vez buscan productos y servicios más eficientes, que les ahorren dinero y que cuiden el medio ambiente. Por todo esto es importante controlar el consumo de energía y tomar acciones para mejorar la eficiencia energética.

Cuando enciendes el horno o la plancha se produce un consumo que normalmente desconoces. Simplemente, a final de mes, recibes la factura de la luz, que te indica el consumo mensual y lo que has de pagar. A muchas de las empresas que tienen procesos industriales les ocurre lo mismo, y es que el control del consumo de manera precisa es algo que no está normalizado en el ámbito doméstico (algunas Smart Home) y en el industrial sigue siendo un aspecto muy mejorable. Este control es muy importante ya que gracias a él podemos mejorar la eficiencia energética, ahorrando dinero para una familia o una empresa y, a la vez, siendo respetuosos con el medio ambiente.

En un estudio del libro Knapp, E., y Samani, R. (2013).*Applied cyber security and the smart grid: Implementing security controls into the modern power infrastructure* se indica que la mejora de la eficiencia energética de la red eléctrica de los Estados Unidos de América en un 5% equivaldría a reducir el impacto climático de 53 millones de coches. Este dato nos muestra la importancia que debemos darle a mejorar la eficiencia energética mundial. EE.UU. consume el 25% de la energía mundial, lo que podría suponer de manera hipotética, que estaríamos hablando de reducir el impacto climático equivalente a aproximadamente 200 millones de coches simplemente con un 5% de reducción del gasto energético global gracias a la mejora de la eficiencia.

Este trabajo busca acercar al lector al uso de la nube y sus tecnologías asociadas para mejorar la eficiencia energética de un edificio, una vivienda, un campus universitario, un centro comercial y muchos más lugares. Además, se muestra un prototipo construido para analizar y predecir el consumo de una vivienda, alertar de posibles consumos inesperados y servir de base para proyectos futuros en el campo de la eficiencia energética los cuales integren herramientas en la nube.

2 Marco Teórico

Para poder entender la situación actual en la que nos encontramos a nivel tecnológico es necesario hablar de ciertos conceptos, explicarlos y comparar distintas herramientas para conocer cuáles nos ofrecen mejores prestaciones para la implementación de soluciones tecnológicas para la mejora de la eficiencia energética, como puede ser el prototipo propuesto en este trabajo.

En primer lugar, debemos definir la eficiencia energética y comentar en qué ámbitos podría mejorarse. Algunos de estos ámbitos hacen referencia a las Smarts Grids o las Smart Home, de las hablaremos más en profundidad después. Finalmente, se explica el estado del arte de las tecnologías en la nube y las herramientas software y hardware que nos ayudarán a cumplir con los objetivos de este proyecto.

2.1 Eficiencia energética

Según Wikipedia, la eficiencia, para la física "es la relación entre la energía útil y la energía invertida". Esto significa que mejorar la eficiencia es minimizar la energía invertida o consumida que se pierda.

Las pérdidas de energía pueden darse en múltiples casos en el hogar como, por ejemplo, cuando dejamos una luz encendida en una habitación en la que no hay nadie o cuando calentamos el horno durante demasiado tiempo antes de cocinar. Casos similares se dan actualmente también en la industria y es necesario realizar un control del consumo y minimizar estas pérdidas para conseguir mejorar la eficiencia.

Los beneficios que obtenemos al mejorar la eficiencia energética según la Agencia Internacional de la Energía, entre otros, son:

1. Ahorro energético
2. Mejores precios de la electricidad
3. Mejor gestión de recursos energéticos
4. Mayor presupuesto público
5. Mitigación de la pobreza energética
6. Empleo
7. Menor contaminación

8. Impacto positivo en el cambio climático

9. Mayor productividad industrial

Solamente por los beneficios mencionados anteriormente merece la pena investigar en este campo, realizar proyectos innovadores y fomentar el uso de herramientas punteras, como es la nube, para acercar estas soluciones a empresas y familias.

Es tan importante para la humanidad mejorar la eficiencia energética que existe un día, el 5 de marzo, denominado "Día mundial de la Eficiencia Energética". Es un sector que está creciendo mucho hoy en día y están surgiendo empresas dedicadas exclusivamente a aportar soluciones energéticas a otras empresas y particulares. También se está invirtiendo mucho dinero en fomentar la creación de estas, así como las que se crean en un sector que tiene mucha relación con el de la eficiencia, como es el de las energías renovables.

Los objetivos de los planes de acción para la eficiencia energética tanto a nivel internacional como nacional buscan "crear una economía de alta eficiencia energética y bajas emisiones de CO2". Los llamados "Cinco Veintes", hacen referencia a que, para el año 2020:

1. Se reduzcan un 20% las emisiones de gases de efecto invernadero

2. Se reduzca un 20% el consumo de energía, promoviendo la eficiencia energética

3. El 20% de la energía ha de proceder de fuentes renovables

2.2 Smart Grid

Las Smart Grids son redes eléctricas inteligentes que incorporan herramientas informáticas y domóticas, así como cualquier tecnología puntera para conseguir una respuesta óptima a la demanda volátil de electricidad. Estas pueden estar asociadas a un edificio, una universidad o una región.

Para entenderlas mejor, cabe mencionar que la electricidad, por lo general, debe consumirse simultáneamente a su producción en la planta eléctrica. Aunque existen muchos proyectos de plantas de almacenamiento para que, si se produce un exceso de producción eléctrica, no se pierda la energía, si no que se guarde para ser posteriormente utilizada cuando sea necesario.

Responder manualmente a las fluctuaciones de la producción y a la demanda de energía es impensable, por ello los sistemas inteligentes son fundamentales. Estos sistemas realizan ajustes en la red de manera automatizada, rápida y eficiente. Asimismo, las Smart Grids cuentan con ventajas además de la eficiencia y de todos los beneficios que esta trae, como son la mayor seguridad de la red, la integración de todo tipo de tecnologías y un control preciso de los consumos.

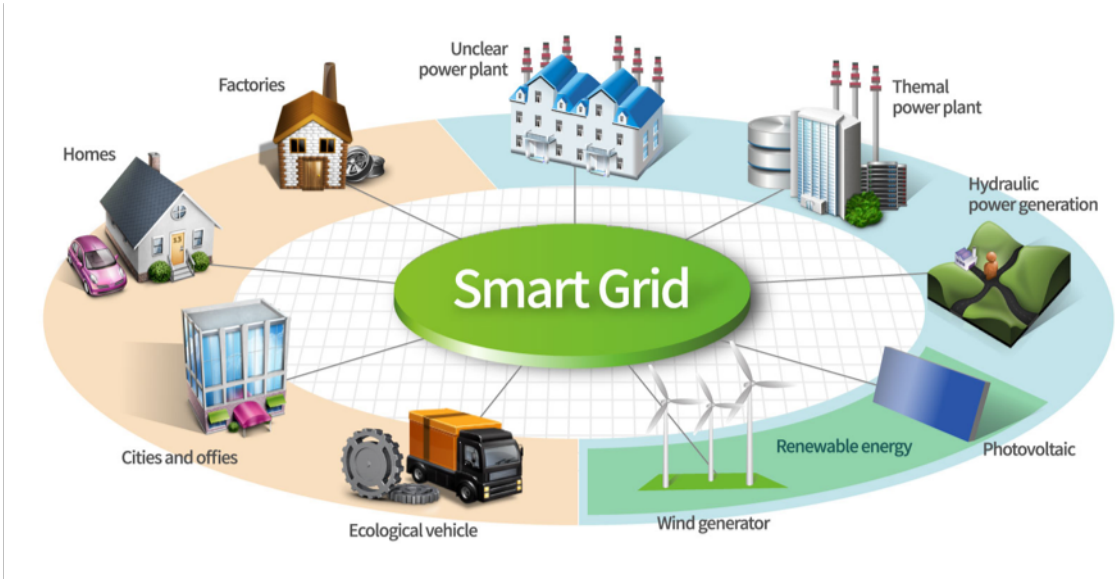


Figura 2.1: Smart grid. Fuente: CENERGIA

2.3 Smart Home

La Smart Home o casa inteligente es un concepto que define la automatización y la conectividad en el ámbito del hogar. La automatización de procesos como cerrar una persiana, encender el aire acondicionado o apagar las luces entran en este concepto, así como la comunicación entre diferentes dispositivos, como cuando un asistente virtual le ordena al robot aspirador que aspire el suelo de una habitación.

Uno de los aspectos más importantes de una Smart Home es el ahorro energético. Una casa inteligente debe tener la capacidad de optimizar el uso de los recursos energéticos, suponiendo un ahorro para las familias y múltiples beneficios para la sociedad.

Utilizan dispositivos como sensores de iluminación, sensores de humedad, sensores de temperatura, micrófonos, altavoces, luces, sensores ópticos, sensores de movimiento y más. El objetivo es obtener la mayor cantidad de datos para que el sistema tenga un conocimiento del entorno lo más completo posible. Gracias a estos datos, obtenemos una información muy valiosa para crear aplicaciones y soluciones inteligentes en una vivienda.

Los datos obtenidos se pueden tratar de manera local, como funcionan la mayoría de las soluciones de Smart Home hoy en día, o pueden tratarse en la nube, aunque sería necesaria la conexión a Internet. Realmente, dependerá de la aplicación, del objetivo de los datos y del volumen de estos, que una aplicación sea mejor en local o haciendo uso de la nube.

Figura 2.2: Ejemplo de aplicación para Smart Home. Fuente: GIRA

2.4 La nube

La nube es una metáfora de internet. Es un término que engloba el almacenamiento, el procesamiento y muchos tipos de servicios online. Cuando hablamos de usar la nube para un proyecto, como este, estamos diciendo que usaremos herramientas que nos ofrecen distintas empresas a través de Internet. Estos pueden ser el almacenamiento de datos, es decir, bases de datos online, ejecución de aplicaciones de manera externa en ordenadores más potentes, comunicaciones de todo tipo, incluso aplicaciones móviles.

La nube, como Internet, tiene un potencial ilimitado y ofrece unas posibilidades enormes a la hora de diseñar soluciones ingenieriles. Por tanto, en el campo de la eficiencia energética creemos es fundamental tenerla en cuenta.

Las ventajas que ofrece la nube con respecto a soluciones locales son:

1. **Escalabilidad.** Es realmente sencillo escalar una aplicación según la demanda, contratar más espacio de almacenamiento si es necesario y velocidad de procesamiento mayor si se requiere. Además, algunos de los proveedores de servicios online ofrecen escalabilidad automática, por lo que el usuario no debe preocuparse de contratar más espacio ni procesadores más potentes.
2. **Velocidad.** Para aplicaciones que requieran de procesadores muy potentes, como las basadas en Machine Learning o Deep Learning, la nube es una solución excelente, ya que no necesitas comprar ordenadores potentes, si no que se “alquilan” estos a través de la nube.
3. **Facilidad de implementación.** La mayoría de los proveedores de estos servicios facilitan al usuario la creación de aplicaciones y soluciones por medio de una interfaz sencilla y tutoriales propios.

4. **Seguridad.** Los proveedores garantizan la seguridad de sus plataformas y servidores e invierten muchísimo dinero en evitar ataques informáticos, por lo que los servidores de la nube, por lo general, serán más seguros que los servidores locales.

5. **Mantenimiento.** De este aspecto también se encargan los proveedores, por lo que es un punto importante para tener en cuenta a la hora de invertir en servidores.

6. **Menor inversión.** Los altos costes y servicios de comprar unos equipos locales no existen. Se paga por el uso de los servidores y los precios suelen ser accesibles para cualquier empresa.

7. **Actualizaciones automáticas.** Los equipos y el software de los proveedores están, por lo general, siempre actualizados. No siempre ocurre esto cuando las empresas compran equipos y, hoy en día que la tecnología avanza tan deprisa, esta es una ventaja enorme.

8. **Accesibilidad.** Es posible acceder a los datos desde cualquier parte del mundo en la que haya Internet. Por lo que abre muchas posibilidades para el trabajo en equipo o la velocidad de reacción frente a cambios, entre otras cosas.

La nube toma aún más importancia, y ventaja frente a las aplicaciones locales, si tenemos en cuenta los progresos en la red inalámbrica global con la instauración de la tecnología 5G, la nueva generación que traerá consigo una revolución tecnológica en todos los ámbitos.

Figura 2.3: Diagrama de la nube.

Hay distintos tipos de nubes:

2.4.1 Nube pública

Es una nube computacional mantenida y gestionada por terceras personas no vinculadas con la organización. En este tipo de nubes tanto los datos como los procesos de varios clientes se mezclan en los servidores, sistemas de almacenamiento y otras infraestructuras de la nube. Los usuarios finales de la nube no conocen qué trabajos de otros clientes pueden estar corriendo en el mismo servidor, red, sistemas de almacenamiento, etc. Aplicaciones, almacenamiento y otros recursos están disponibles al público a través del proveedor de servicios, que es propietario de toda la infraestructura en sus centros de datos. El acceso a los servicios solo se ofrece de manera remota, normalmente a través de internet. Dentro de estas nubes podemos encontrar englobadas las Cloud Privado virtual consideradas como nubes de dominio público pero que mejoran la seguridad de los datos. Estos datos se encriptan a través de la implantación de una VPN.

2.4.2 Nube privada

Son una buena opción para las compañías que necesitan alta protección de datos y ediciones a nivel de servicio. Las nubes privadas están en una infraestructura bajo demanda, gestionada para un solo cliente que controla qué aplicaciones debe ejecutarse y dónde. Son propietarios del servidor, red, y disco y pueden decidir qué usuarios están autorizados a utilizar la infraestructura. Al administrar internamente estos servicios, las empresas tienen la ventaja de mantener la privacidad de su información y permitir unificar el acceso a las aplicaciones corporativas de sus usuarios.

2.4.3 Nube híbrida

Combinan los modelos de nubes públicas y privadas. Un usuario es propietario de unas partes y comparte otras, aunque de una manera controlada. Las nubes híbridas ofrecen la promesa del escalado, aprovisionada externamente, a demanda, pero añaden la complejidad de determinar cómo distribuir las aplicaciones a través de estos ambientes diferentes. Las empresas pueden sentir cierta atracción por la promesa de una nube híbrida, pero esta opción, al menos inicialmente, estará probablemente reservada a aplicaciones simples sin condicionantes, que no requieran de ninguna sincronización o necesiten bases de datos complejas. Se unen mediante la tecnología, pues permiten enviar datos o aplicaciones entre ellas. Un ejemplo son los sistemas de correo electrónico empresarial.

2.4.4 Nube comunitaria

De acuerdo con Joyanes Aguilar, en 2012, el Instituto Nacional de Estándares y Tecnología de los Estados Unidos de América define este modelo como aquel que se organiza con la finalidad de servir a una función o propósito común, como política o seguridad, las cuales son administradas por las organizaciones constituyentes o terceras partes.

2.5 Tecnología 5G

Se prevé la creación de un ecosistema masivo basado en el Internet de las Cosas gracias a la nueva generación de redes de telecomunicaciones. Esta revolución ya ha comenzado este año 2019 y se están comercializando dispositivos adaptados a esta tecnología. Empresas tecnológicas punteras compiten por liderar esta revolución.

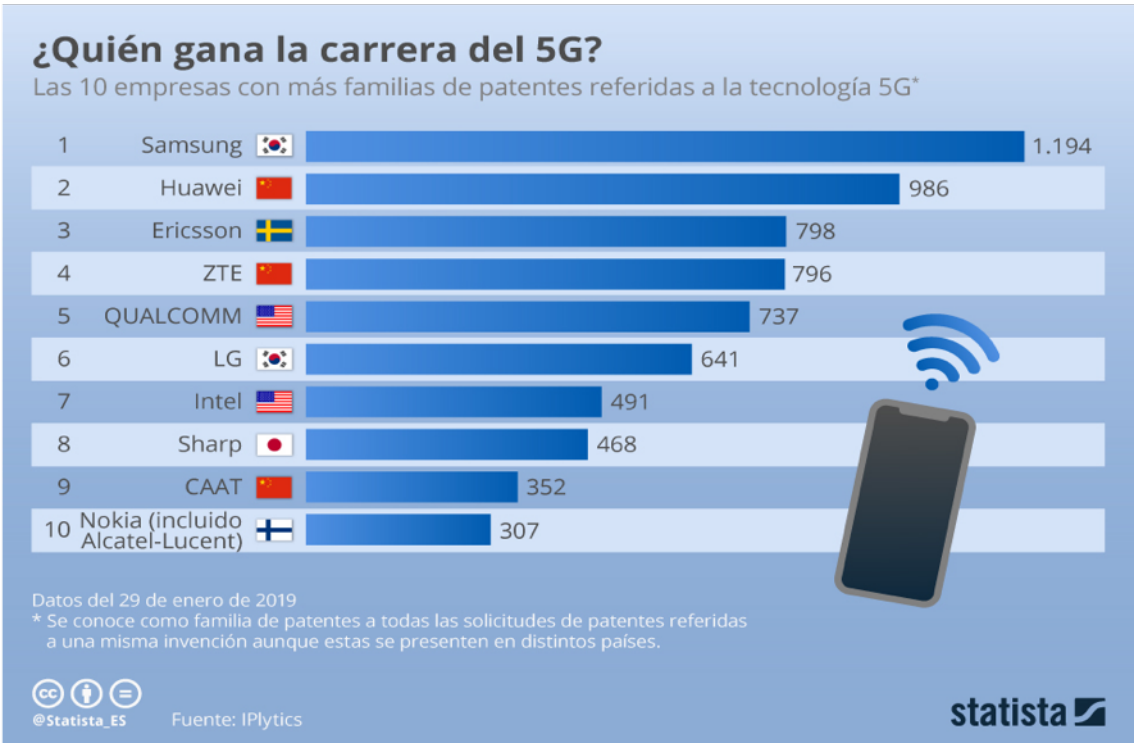


Figura 2.4: Gráfico de las empresas líderes en tecnología 5G. Fuente: IPlytics

Las mejoras de esta generación con respecto a la generación anterior son múltiples:

1. **Velocidad.** Hasta 10 Gbps, una mejora de 10 a 100 veces mejor que el 4G y el 4.5G
2. **Latencia.** Con 1 milisegundo, es apta para las aplicaciones en tiempo real como los vehículos autónomos.
3. **Ancho de banda.** Hasta 1000 veces más rápida por unidad de superficie.
4. **Cobertura.** Aseguran una cobertura del 100%.
5. **Consumo eléctrico.** Ahorro del 90% en el consumo de energía de la red.
6. **Durabilidad** Hasta 10 años de duración de las baterías de dispositivos IoT (Internet of Things).

Se pronostica que el 5G sea la generación más rápida en implementarse a escala global, cubriendo un 40% de la población y 1500 millones de suscripciones para el año 2024. Además, para 2020 habrá más de 20 mil millones de dispositivos conectados al Internet de las Cosas. Esto abre un nuevo mundo y oportunidades para las empresas para crear soluciones y aplicaciones basadas en la nube, ya que serán rápidas, seguras, duraderas y en tiempo real.

2.6 Servicios en la nube (Servicios Cloud)

Debemos diferenciar 3 tipos de servicios ofrecidos por los proveedores a través de Internet, aunque en la práctica a veces no exista una línea clara entre un servicio u otro. Estos son:

2.6.1 IaaS

Las siglas hacen referencia a Infrastructure as a Service (Infraestructura como Servicio). Permite al usuario o empresa virtualizar todo el hardware. El cliente por tanto se encarga de todo lo relativo a la programación. Las interfaces suelen ser complejas.

2.6.2 PaaS

Las siglas hacen referencia a Platform as a Service (Plataforma como Servicio). El proveedor ofrece una plataforma con una interfaz sencilla para crear aplicaciones y procesar y almacenar datos.

2.6.3 SaaS

Las siglas hacen referencia a Software as a Service (Software como Servicio). Con este servicio el cliente paga directamente por las aplicaciones en la nube, como podría ser la suscripción a una aplicación de móvil, por ejemplo.

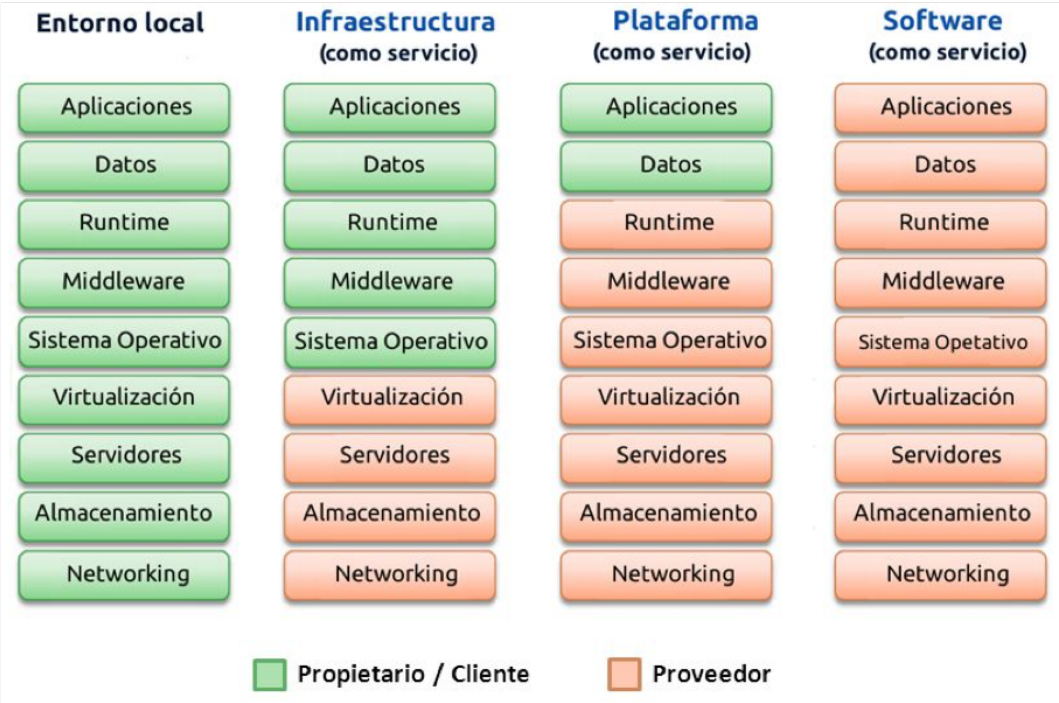


Figura 2.5: Comparación de las herramientas ofrecidas por los distintos tipos de servicios Cloud y las soluciones locales. Fuente: J. Fernández (2018). *Implantación de una solución de escritorios virtuales con OpenStack.*

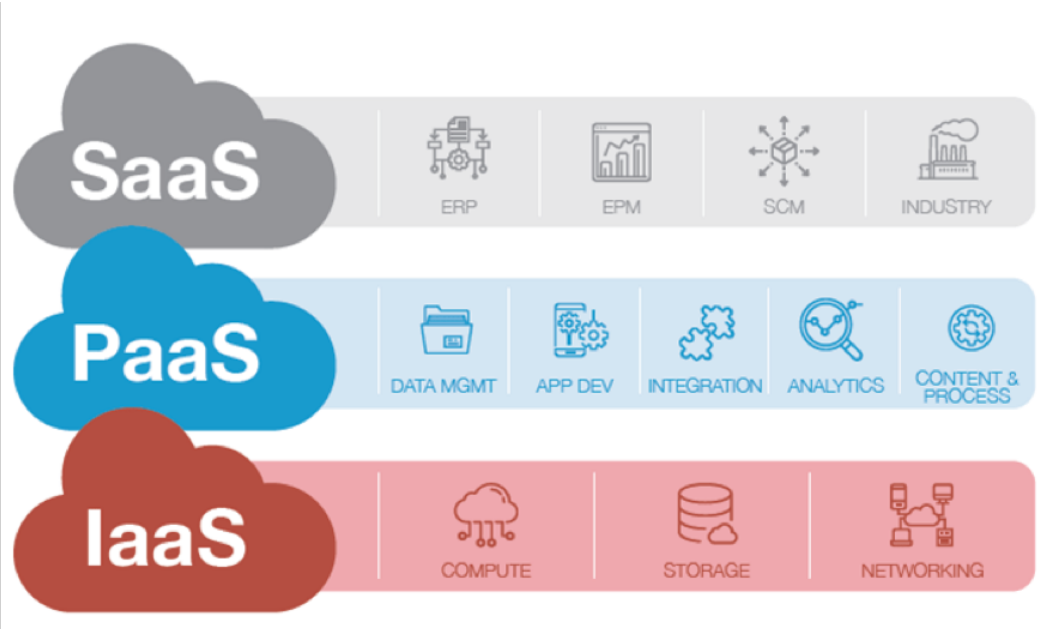


Figura 2.6: Comparación de las herramientas ofrecidas por los distintos tipos de servicios Cloud. Fuente: Neteris

Los servicios en la nube los realizan multitud de empresas, pero el mercado lo ocupan gigantes tecnológicos en su mayor parte, como podemos ver en el siguiente gráfico:

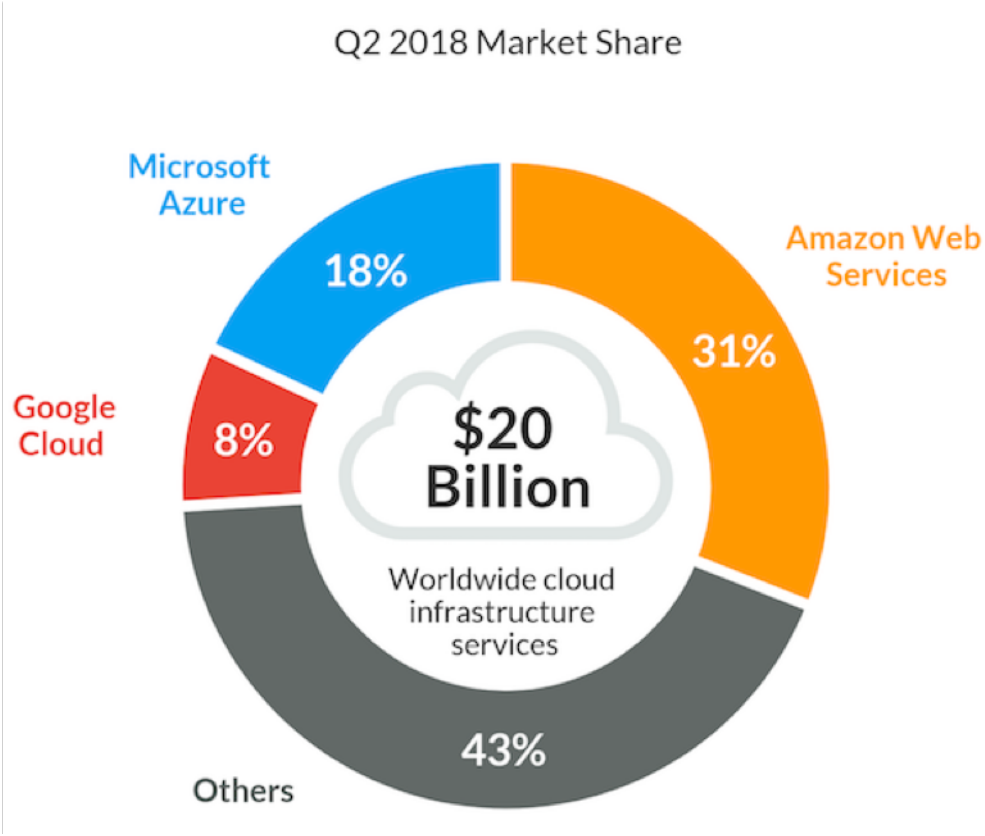


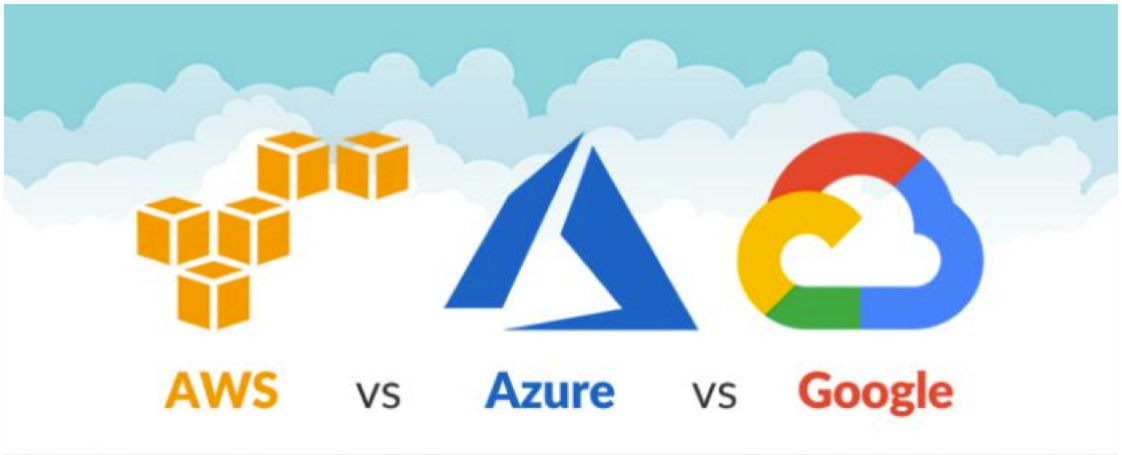
Figura 2.7: Cuota de mercado de los servicios en la nube. Fuente: Canalys

Destacamos a Amazon Web Services por su cuota de mercado, que ya adelantamos, será el proveedor que utilizaremos en la nube para realizar el prototipo de este trabajo.

Las 5 plataformas con más importancia en el mercado son, la ya mencionada, AWS, Microsoft Azure, Google Cloud Platform, IBM Bluemix y Alibaba Cloud. Existen plataformas gratuitas como KAA, Ubidots o VMWare, pero, obviamente, no consiguen la capacidad de procesamiento, almacenamiento, ni la cantidad de servicios que ofrecen las plataformas de pago. Además, las opciones de capa gratuita que ofrecen las plataformas de pago hacen la competencia directa a las plataformas gratuitas y, además, sirven para aprender a usar plataformas potentes de pago y simular aplicaciones que tienen mucho más potencial de negocio. Esta es una de las razones por las que se ha elegido utilizar la capa de Amazon Web Services para la construcción del prototipo.

Para la elección de una plataforma tendremos en cuenta aspectos tecnológicos, administrativos, comunicativos, de compatibilidad y económicos, entre otros. A través del enlace `http://comparecloud.in/?_lrsc=c276bb2a-7ec4-4a23-8f51-9ac3645823b8` el usuario

puede consultar los servicios que ofrecen las plataformas más importantes, las top 5 mencionadas más la plataforma de Oracle. La cantidad de servicios es tan grande que se ha decidido añadir el enlace en lugar de enumerar cada uno de los servicios. La comparación que podemos realizar de una manera más sencilla es la que nos ha ofrecido Whizlabs entre las 3 principales plataformas en la nube:



PROVEEDORES DE SERVICIOS EN LA NUBE		
Amazon Web Services	Microsoft Azure	Google Cloud
EXPERIENCIA EN EL SECTOR		
12 años	7 años	6 años
¿POR QUÉ ELEGIR ESTAS PLATAFORMAS?		
Domina el mercado en configuración, monitorización, seguridad y otros. Servicios maduros y extensos. Abierto y flexible. Alcance global. Servicio al cliente excelente. Tutoriales.	Integración con herramientas Microsoft. Amplio conjunto de características. Líder en desarrollo y pruebas de herramientas. Soporte open source. Nube híbrida.	Soporte open source y portabilidad. Contratos flexibles y descuentos. Diseñada para negocios basados en la nube. Facilidad para el desarrollo de software (metodología DevOps).
¿POR QUÉ NO ELEGIRLAS?		
Difícil de usar. Sobrecarga de opciones. Administración de los costes.	Herramientas de administración poco eficientes. Menos preparado para la empresa.	Reciente entrada en el mercado de servicios en la nube. Menos centros de datos en el mundo. Menos servicios y características.
CUOTA DE MERCADO		
62%	20%	12%
FIJACIÓN DEL PRECIO (Las tres plataformas siguen un modelo de cobro por uso calculado automáticamente)		
Por hora	Por minuto	Por minuto

Figura 2.8: Comparación de los proveedores de servicios en la nube más importantes. Fuente: Whizlabs

Como se ha comentado anteriormente, la elección de una de estas plataformas se ha basado en la cuota de mercado, la cantidad de servicios que ofrecía la capa gratuita, los tutoriales que se ofrecían sobre estos servicios y la posibilidad de aprender sobre la plataforma más usada hoy en día por empresas de todo el mundo. Por ello Amazon Web Services ha sido la plataforma escogida para la realización de un prototipo en este trabajo.

2.7 Bases de datos

Una de las partes fundamentales para construir aplicaciones son las bases de datos. Estas nos permiten guardar y consultar posteriormente un historial de lecturas, ya sea de sensores, de interacciones con la aplicación o de comandos ejecutados; depende de lo que queramos y más nos interese guardar.

Para cada aplicación en específico necesitamos elegir la base de datos más adecuada de entre los tipos que existen:

2.7.1 Bases de datos jerárquicas

Estas se organizan en forma de árbol invertido. El primer nodo, que no tiene padres, es el nodo raíz y los nodos que no tienen hijos son nodos hoja.

2.7.2 Bases de datos de red

Son similares a las jerárquicas, la diferencia es que los nodos pueden tener varios padres y no es necesario que haya nodo raíz y nodos hoja, formando de esta manera una red.

2.7.3 Bases de datos transaccionales

La característica principal de estas es su velocidad. Se usan para garantizar el envío de datos, por lo que la redundancia y duplicidad no es un problema. Suelen conectarse de alguna manera a una base de datos relacional.

2.7.4 Bases de datos relacionales

Es el modelo más usado en la actualidad para administrar datos dinámicamente. Se basa en relaciones entre los datos. Asociando de esta manera unos datos a otros. El lenguaje más habitual es SQL, que se usará en este trabajo para enviar y extraer datos de la base de datos.

2.7.5 Bases de datos multidimensionales

No se diferencian mucho de las bases de datos relacionales. La diferencia es más conceptual. Los campos de una base dimensional pueden ser de dos tipos, representar dimensiones

distintas, momentos distintos o incluso objetivos a alcanzar.

2.7.6 Bases de datos orientadas a objetos

Es un concepto en el cual las leyes de la POO (Programación Orientada a Objetos) se aplican a una base de datos, es decir: herencia, polimorfismo y encapsulación.

2.7.7 Bases de datos documentales

Permiten indexar un texto completo y así realizar búsquedas potentes con grandes volúmenes de información.

2.7.8 Bases de datos deductivas

Permiten hacer deducciones a través de inferencias. Se llaman también bases de datos lógicas. Según ciertas reglas encontraremos cierta información y podremos añadir información que se traducirá en un cambio en las reglas.

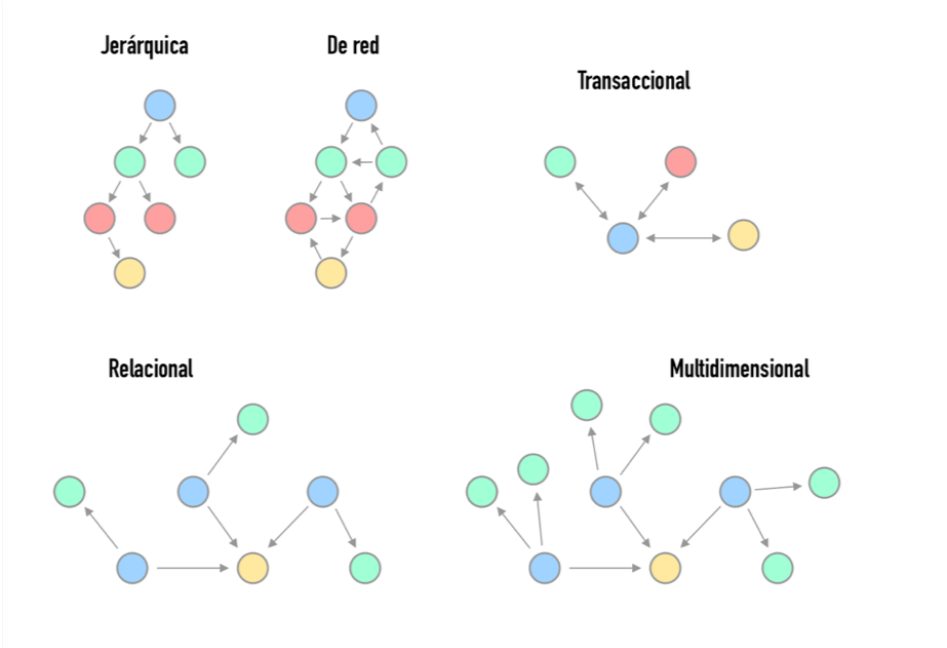


Figura 2.9: Tipos de bases de datos.

Existe otra clasificación muy común en el sector de las bases de datos. Diferencia las bases de datos relacionales de las que no lo son. Concretamente, se suele hablar de SQL o NoSQL.

Una base de datos no relacional (NoSQL) tiene una estructura descentralizada, donde los datos no están ordenados. Tenemos una escalabilidad horizontal, ya que puedes añadir distintas fuentes de datos, como sensores y que la base de datos siga funcionando correctamente. El uso de estas bases de datos está muy enfocado al tiempo real y a aplicaciones como videojuegos, Internet de las Cosas o redes sociales.

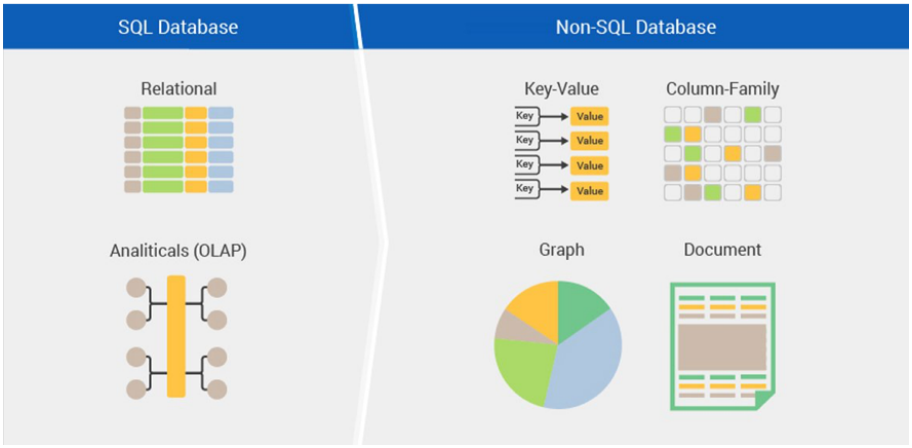


Figura 2.10: SQL vs. NoSQL. Fuente: Net Solutions

En una base de datos de Key →Value, por ejemplo, a cada Key se le asocia un valor, pero esta base de datos no está ordenada, si no que sus datos se encuentran en un Data Lake. Cuando tenemos lecturas de varios sensores para una aplicación de IoT, ocurre algo similar, ya que en el caso de las NoSQL, no se guarda el historial como tal de las lecturas, sino que tienes un Data Lake con los valores de los sensores en ese instante. Estas aplicaciones utilizan bases de datos NoSQL.

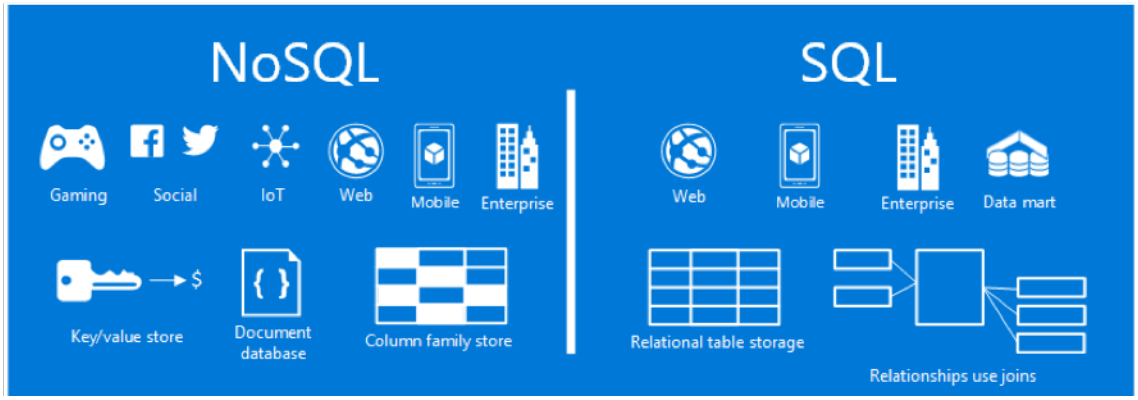


Figura 2.11: Aplicaciones SQL vs. NoSQL. Fuente: Sena. M. (2017). *Cómo pasar de SQL a NoSQL sin sufrir.*

Una vez sabemos en qué consisten las bases de datos y qué tipos existen, podemos elegir

una base de datos para una aplicación. Aquí entramos en elegir una solución local o un servicio a través de Internet. Por lo que hemos estado viendo, para el prototipo de este trabajo elegiremos una base de datos en la nube, lo que nos dará escalabilidad y compatibilidad con otros servicios.

Los proveedores de servicios de bases de datos más solicitados que existen actualmente son:

SQL → Microsoft SQL Server, PostgreSQL, MySQL, SQLite, Firebird, Oracle…

NoSQL → MongoDB, Apache Cassandra, Amazon DynamoDB, Oracle NoSQL…

Figura 2.12: Proveedores más importantes de servicios de bases de datos. Fuente: Google

Al elegir una base de datos para nuestra aplicación debemos tener en cuenta que debe ser compatible con la plataforma elegida, con los dispositivos, que sepamos usar correctamente gracias a nuestros conocimientos y la información disponible en Internet y que sea posible crear la base de datos dentro de los límites del presupuesto.

2.8 Comunicaciones en el Internet de las Cosas (IoT)

La comunicación es uno de los aspectos fundamentales de un proyecto del Internet de las Cosas. Para comunicar datos desde los dispositivos a la nube, de esta a los dispositivos y estos entre ellos, existen diferentes tecnologías y protocolos.

Para poder utilizar la nube, es necesaria la conexión a Internet. Esta la podemos obtener mediante un ISP (Proveedor de servicios de Internet), que nos proporcionará acceso a la red mediante tecnologías como el 5G o fibra óptica.

La comunicación a los puntos de acceso a Internet y entre los distintos dispositivos en este tipo de aplicaciones suele ser inalámbrico. Existen numerosas tecnologías que se utilizan hoy en día para la comunicación inalámbrica. Tres de las tecnologías más utilizadas son: Wi-Fi, Bluetooth, ZigBee. Existen otras tecnologías que están surgiendo actualmente como LoRa o

NarrowBand IoT que ofrecen características interesantes en cuanto a eficiencia energética.

El estándar Wi-Fi se considera una tecnología para la comunicación inalámbrica de área local, mientras que el Bluetooth y el ZigBee son considerados de área personal (por su alcance, velocidad, consumo…). También tenemos las tecnologías como el 4G o el 5G, que pueden usarse como tecnología de comunicación, aunque dependen de los ISP. Esta última tecnología puede utilizarse para la conexión de dispositivos móviles con Internet o aquellos dispositivos que incorporen el 5G en un futuro. Esta es la última apuesta por las compañías de telecomunicaciones, un sistema M2M (Machine to Machine), basada en el modelo de negocio del GPRS. Se podría decir que, por su consumo, coste y escalabilidad, esta tecnología es un enfoque completamente diferente al resto.

En el diseño de aplicaciones IoT actuales, la conexión de controladores, sensores e Internet suele realizarse mediante conexiones de área local/personal de bajo consumo. Las más utilizadas son las que hemos comentado anteriormente. A continuación, se muestra una tabla comparativa de las características de cada tecnología.

TECNOLOGÍA	CONSUMO	ALCANCE	MADUREZ	DISPONIBILIDAD	SEGURIDAD	USABILIDAD	TASA DE DATOS
GSM/GPRS	Muy alto	Alto	Muy Alto	Muy alto	Alta	Alta	Alta
SigFox	Bajo	Medio	Alto	Medio	Media	Alta	Muy baja
LoRa	Bajo	Medio	Bajo	Muy bajo (ad hoc)	N A	Baja	Muy baja
NB IoT							
WiFi	Alto	Bajo	Muy alto	Alto	Baja	Alta	Muy alta
BLE	Muy bajo	Muy bajo	Alto	Bajo	Baja	Media	Baja
ZigBee	Medio	Bajo	Medio	Muy bajo	Alta	Baja	Baja

Figura 2.13: Comparación de las tecnologías de comunicación más utilizadas en IoT. Fuente: Efor

Estas características hacen de ZigBee la tecnología más utilizada para dispositivos de Smart Home. No permite enviar una cantidad de datos grande (tampoco es necesario en la mayoría de las aplicaciones de IoT), pero el consumo de los dispositivos que lo utilizan es muy bajo, pudiendo durar las baterías de estos hasta 10 años.

Las características principales de una red de comunicaciones IoT son:

1. **Conexiones bidireccionales seguras**
2. **Bajo consumo de energía**
3. **Largo alcance de comunicación**
4. **Bajo volumen de envío datos**

Hay otras características que son específicas de cada aplicación, ya sea de tiempo real, de

análisis de datos, de control y automatización, etc.

Para aplicaciones relacionadas con la eficiencia energética, debemos pensar primero en cual es el objetivo y la naturaleza de la aplicación. Si el objetivo es analizar y predecir el consumo de una vivienda, las comunicaciones que utilizaremos puede que no sean las mismas que si nuestro objetivo es controlar la iluminación, la calefacción, etc.

2.9 Protocolos de comunicación en IoT

Dentro de las tecnologías que se usan en el Internet de las Cosas, tenemos protocolos de comunicación. La gran mayoría de estos protocolos utilizan un modelo Publisher/Subscriber. Es decir, los que envían los mensajes (publishers) envían el mensaje sin tener un receptor específico. Publican el mensaje y los subscribers eligen qué mensaje recibir, según su interés u objetivo.

Los principales protocolos de comunicación son:

2.9.1 HTTP

A priori, sus características no lo hacen ideal para un sistema IoT. Pero es uno de los protocolos más estandarizados y se usa, por ejemplo, para insertar datos en una base de datos SQL.

2.9.2 MQTT

Es el protocolo ideal para el control de una red con muchos dispositivos. Es fácil de implementar, pero consume bastante debido a que está basado en TCP, y tiene una seguridad limitada. Se caracteriza por ofrecer una calidad de servicio excelente.

2.9.3 DDS

Es muy parecido a MQTT, pero conecta los dispositivos directamente entre ellos, sin usar un servidor. Según la aplicación puede ser interesante, pero el servidor ofrece potentes herramientas de análisis y procesamiento de datos que están limitados con este protocolo.

2.9.4 XMP

Usa un formato XML para los mensajes e identifica a cada dispositivo de la red con una dirección JID (Jabber ID, que sigue el estándar nombre@dominio.com). Está basado en TCP y utiliza una arquitectura cliente-servidor.

2.9.5 AMQP

Se centra en la seguridad y en garantizar que los mensajes lleguen de un servidor a otro (a pesar de apagones o fallos). La principal contra la encontramos en el consumo.

2.9.6 JMS

Es una API dentro de la plataforma Java para enviar mensajes entre dos aplicaciones o dispositivos (como veremos, Java es uno de los principales lenguajes de programación en IoT).

Dentro del IoT Core de Amazon Web Services se utiliza MQTT. Por lo que es el protocolo en el que más hincapié se ha hecho en este trabajo. También es el protocolo más utilizado en el Internet de las Cosas y más información podemos encontrar en Internet.

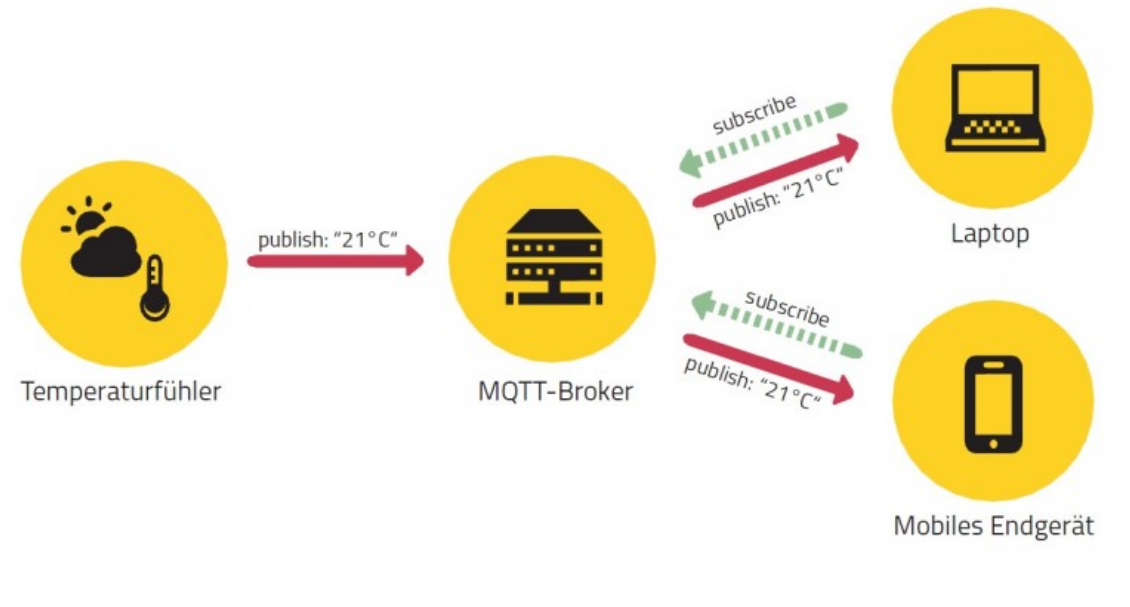


Figura 2.14: Protocolo MQTT. Fuente: Mai, Marc. (2016) *MQTT für Dummies.*

2.10 Herramientas Software

Las herramientas software que podemos utilizar para una aplicación de esta naturaleza son muchas. Nos centraremos en comentar qué tipo de herramientas pueden ser necesarias para este trabajo y en realizar una explicación de las que usaremos.

Al usar una plataforma online simplificamos mucho el uso de software y hardware. Simplemente debemos usar las herramientas que nos ofrece la plataforma y complementar algunas funcionalidades con el software que queramos. De hecho, una de las ventajas de estas plataformas es que son compatibles con software externo en muchos campos.

Como es lógico, necesitamos un navegador web para utilizar la plataforma. No comentaremos mucho más sobre navegadores ya que es un aspecto poco esencial del trabajo y casi todos cumplen bien, así que cada usuario puede usar el que mejor se ajuste a sus gustos.

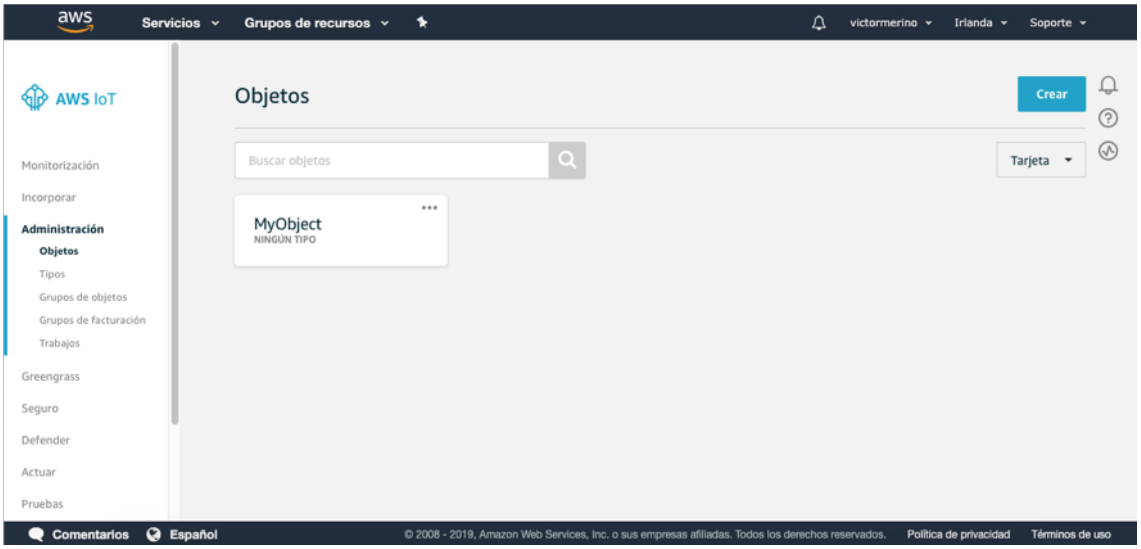


Figura 2.15: IoT Core de AWS.

La conexión entre distintos puntos de una aplicación como la plataforma, los dispositivos IoT o la base de datos utiliza protocolos de comunicación anteriormente comentados. Para el uso de estos protocolos se suelen utilizar aplicaciones que implementen estos protocolos y ejecutables de lenguajes de programación con librerías que los implementen.

2.10.1 Herramienta para bases de datos

Un ejemplo de herramienta para el diseño de bases de datos es MySQLWorkbench, la cual se ha usado en este trabajo para modificar configuraciones de la base de datos SQL, hacer consultas, borrar los datos, etc. Esta herramienta es muy útil ya que utilizas el protocolo SQL, pero no es necesario saberlo. Es una interfaz que facilita en gran medida la manipulación de bases de datos de MySQL.

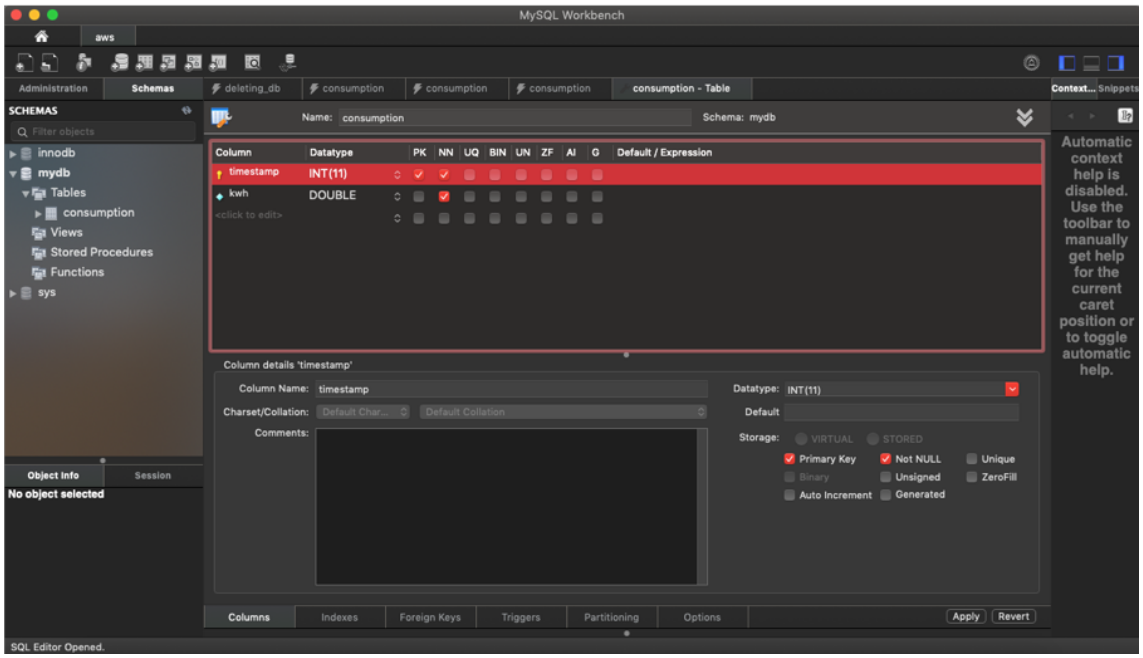


Figura 2.16: Ventana de configuración de la tabla de consumo en MySQLWorkBench.

Como estas herramientas existen otras que sirven para configurar las bases de datos de otros proveedores, como pgAdmin para PostgreSQL. Si elegimos, por ejemplo, la base de datos propia de Amazon (Aurora), la herramienta se utilizará a través del navegador, mediante la propia plataforma de AWS.

También es posible modificar las bases de datos a través del lenguaje SQL usando aplicaciones programadas en Python, C++, MatLab, etc. De hecho, para este trabajo, la inserción de datos de consumo en la base de datos de MySQL se realiza a través de un script de Python que utiliza una librería de SQL.

2.10.2 Lenguajes de programación

Los lenguajes de programación más usados en IoT según una encuesta de la IoT Developer Survey 2018 son, por orden:

Dispositivos	Puertas de enlace	Nube
C	Java	Java
C++	Python	Javascript
Python	C	Python
Java	C++	PHP

Figura 2.17: Lenguajes de programación más utilizados en IoT. Fuente: IoT Developer Survey (2018)

2.11 Dispositivos para el prototipado de sistemas IoT

Cualquier dispositivo capaz de conectarse a Internet, enviar y/o recibir información se puede considerar un dispositivo del Internet de las Cosas. Estos dispositivos, haciendo uso de las tecnologías que hemos comentado anteriormente (M2M, Wi-Fi, ZigBee...) se comunican entre ellos y con la nube para lograr ciertas funciones. Por ejemplo, un sensor de temperatura que cuenta con un microcontrolador y comunicación Bluetooth LE se considera un dispositivo IoT, ya que es capaz de comunicar las lecturas de temperatura a una red WPAN para enviar estos datos a la nube o a otro dispositivo para su procesamiento.

A la hora de diseñar una aplicación de IoT, debemos elegir unos dispositivos que cumplan los requisitos técnicos (compatibilidad, velocidad, duración...) y presupuestarios. En soluciones sencillas como proyectos académicos se suelen utilizar controladores complejos que faciliten la programación, la conexión a Internet y las conexiones con los sensores y otros dispositivos. Para proyectos más complejos y presupuestos más elevados se plantean los aspectos de diseñar dispositivos específicos para la aplicación.

En este trabajo se estudian las opciones para soluciones sencillas ya que nos encontramos con un escenario de investigación y desarrollo a nivel académico. Es decir, se utilizará un dispositivo avanzado como la Raspberry Pi de base para el proyecto. Se usa para realizar la conexión a Internet, tratamiento de datos, comunicación con la nube, conexión con los sensores, etc.

La Raspberry es la opción idónea para este tipo de proyectos por su calidad-precio, las tecnologías adecuadas para proyectos IoT de las que dispone, la cantidad de documentación disponible y los conocimientos adquiridos en el Grado en Ingeniería Robótica.

Otros dispositivos similares para este tipo de proyectos de prototipado son: Arduino Uno, Intel Edison, Udoo Neo, LightBlue Bean, Adafruit Flora, Tessel, Particle Photon, Mediatek Linkit one o C.H.I.P., entre otros.

Figura 2.18: Arduino Uno, izquierda; Raspberry Pi 3 B, centro; Intel Edison, derecha. Fuente: Google

2.12 Sensores

Las aplicaciones IoT se basan en datos. La principal fuente de estos datos son los sensores. Estos pueden ser de muchos tipos, pero en el ámbito de la gestión energética tenemos principalmente:

2.12.1 Sensores de corriente

Pueden ser invasivos o no invasivos. Dan el valor de la corriente que circula por una línea eléctrica. De esta manera se puede medir el consumo de aquello que alimente dicha línea (electrodomésticos, vivienda, centro comercial, etc.).

Figura 2.19: Sensor de corriente no invasivo. Fuente: Google

2.12.2 Sensores de temperatura

La temperatura afecta mucho al consumo de una vivienda por lo que es fundamental controlar la temperatura de los espacios en los que se produzca el consumo y la temperatura exterior.

2.12.3 Sensores de iluminación

Es necesario medir la cantidad de luz que hay en algunos espacios para poder actuar sobre las luces que se encuentren en estos, de manera que se consiga una iluminación correcta con un consumo mínimo.

3 Objetivos del proyecto

Los objetivos del proyecto son:

3.1 Analizar los servicios y las herramientas en la nube

Conocer las herramientas con las que contamos a través de Internet para realizar aplicaciones eficientes energéticamente es fundamental para la gestión energética en los próximos años. Estas permiten desarrollar algoritmos de control y automatización de procesos dentro de los entornos industrial y doméstico de una manera eficiente.

3.2 Analizar los protocolos de comunicación en el Internet de las Cosas

Es necesario conocer cómo comunicar los datos a la nube, las distintas formas de hacerlo y cual es la ideal para una aplicación en concreto.

3.3 Diseñar un sistema de monitorización energética

En este trabajo se plantea un prototipo de monitorización energética para el consumo de una vivienda. En trabajos futuros se comenta la posibilidad de incluir la gestión de la producción de energía renovable dentro del prototipo.

3.4 Desarrollar un prototipo funcional

Para demostrar la funcionalidad del sistema planteado, se ha desarrollado un prototipo de sistema de monitorización energética en el ámbito doméstico haciendo uso de la nube y de las herramientas que esta ofrece y que hemos estudiado previamente.

4 Metodología

4.1 Metodología de desarrollo

La metodología de desarrollo del trabajo es en espiral. Se ha hecho un estudio de las tecnologías disponibles para llevar a cabo el proyecto y se han fijado los objetivos de este. Posteriormente se han estudiado distintos métodos, se ha hecho un diseño y se ha construido un prototipo funcional. El prototipo es el punto de partida para la siguiente iteración de fijación de objetivos, análisis, desarrollo y pruebas.

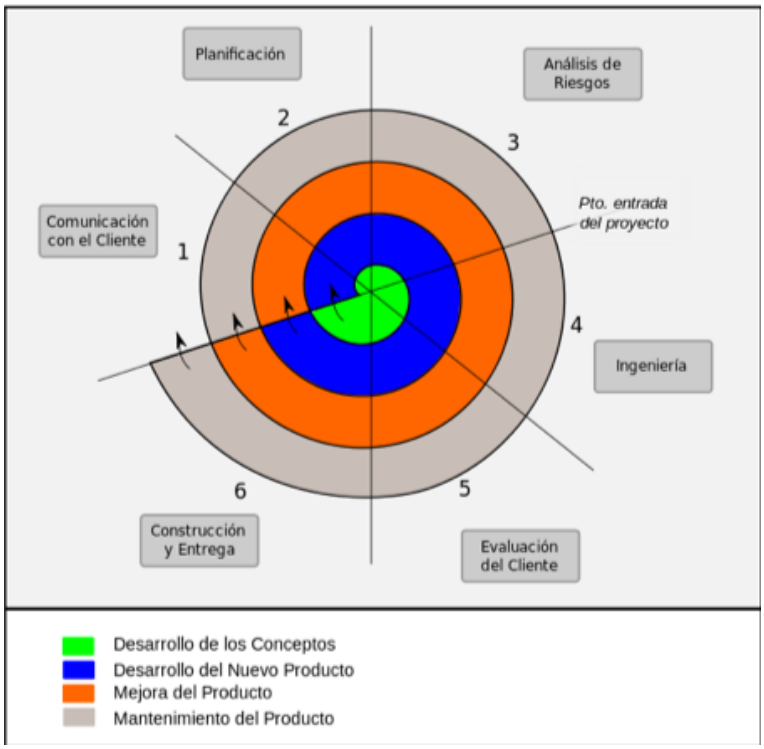


Figura 4.1: Fases de la metodología de desarrollo en espiral. Fuente: Wikipedia

4.2 Hardware

El hardware utilizado en este proyecto se divide en tres apartados:

4.2.1 Raspberry Pi 3 B

La conocida Raspberry Pi 3 Modelo B es un ordenador de placa reducida y bajo coste con sistema operativo Linux. Se utiliza principalmente para la enseñanza y el prototipado (como en este trabajo). Con ella podemos simular un sistema embebido de una manera sencilla gracias a su tamaño y las herramientas que ofrece de programación y conectividad. Dispone de conexión Wi-Fi que nos permite la conexión con la nube. En ella podemos ejecutar facilmente programas en Python.

Su precio ronda los 35€. Este dispositivo cuenta con funciones muy avanzadas que en un sistema real puede que no fueran necesarias. Por ello, podríamos encontrar sistemas embebidos mucho más económicos. Se calcula que para una placa con las funcionalidades requeridas para una aplicación de este tipo su precio mínimo ronde los 10€.

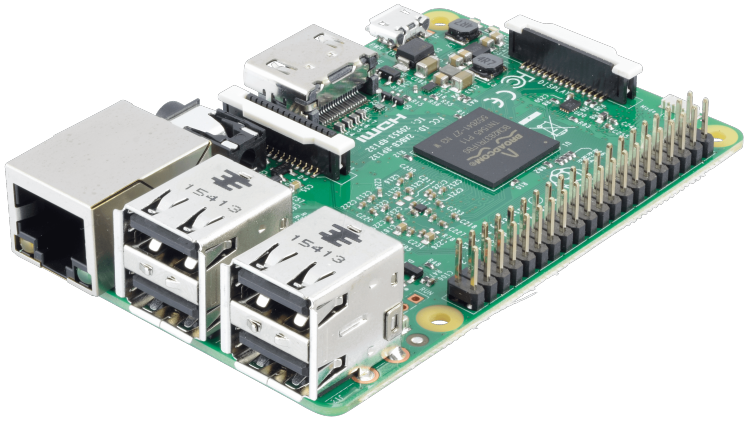


Figura 4.2: Raspberry Pi 3 B. Fuente: Reichelt

4.2.2 Sensor de corriente

El sensor es un AT100 B10 de la marca LEM. Tiene una corriente máxima de lectura de 100 amperios y una tensión de salida que varía de 0 a 10 voltios, proporcional a la potencia leída. Su precio ronda los 60€. Aunque existen opciones más económicas, este ha sido prestado para el trabajo por mi tutor.

Figura 4.3: Sensor de corriente no invasivo AT100B10. Fuente: Google

4.2.3 Convertidor ADC

El convertidor analógico-digital que se utiliza en este trabajo es el Plus Shield de SunFounder. Este se compone de una placa con entradas y salidas que se instala directamente encima de la Raspberry. Es necesario realizar un proceso para la instalación y configuración, además de descargar unas librerías de Python para usarlo. Toda la información de este proceso se encuentra en la web: `http://wiki.sunfounder.cc/index.php?title=Plus_Shield`

Figura 4.4: Convertidor ADC Plus Shield de Sun Founder. Fuente: Google

4.3 Software

4.3.1 Proveedores de servicios Cloud

4.3.1.1 Plataforma

El proveedor elegido para este trabajo ha sido Amazon Web Services. La elección ha sido por cinco principales razones:

1. AWS ocupa la mayor parte del mercado de los servicios en la nube.
2. Es el proveedor que más variedad de herramientas ofrece.
3. La interfaz es relativamente sencilla.
4. La documentación existente en Internet sobre el uso de la plataforma es abundante.
5. Personalmente, me pareció la opción más interesante desde el punto de vista académico y profesional.

4.3.1.2 Bases de datos

Al utilizar AWS como plataforma, necesitamos proveedores de servicios de almacenamiento de datos compatibles con esta. En este trabajo se usan bases de datos SQL y NoSQL. Para cada tipo tenemos un proveedor:

SQL → MySQL

NoSQL → DynamoDB

Ambos son servicios que se pueden usar directamente desde la plataforma de AWS, lo que facilita mucho su uso y compatibilidad con las distintas herramientas de esta.

4.3.2 Lenguajes de programación y librerías utilizadas

4.3.2.1 Python

Por la posición que ocupa Python dentro del sector IoT y los conocimientos adquiridos durante el Grado, se ha elegido lenguaje principal de programación para el prototipo de este trabajo. Contamos con numerosas librerías en Python que nos permiten programar aplicaciones con una compatibilidad desde los dispositivos hasta la nube. Es importante tener en cuenta que un proyecto puede requerir programación en varios lenguajes. Las librerías de Python utilizadas en este trabajo son:

pymysql → Sirve para realizar conexiones con bases de datos MySQL, hacer consultas, borrar datos, etc.

SunFounder_PiPlus.Shield → Es necesaria para la utilización del convertidor ADC. Nos permite leer datos de los pines analógicos.

os → Se utiliza para borrar ficheros. Es una librería que nos permite utilizar funciones del sistema operativo.

time → Se usa para obtener medidas de tiempo. Sirve para registrar el instante en el que se obtiene una medida de un pin, por ejemplo.

RPi.GPIO → Librería necesaria para el uso de los pines de la Raspberry.

glob2 → Tiene una función para la lectura de los nombres de los ficheros de un directorio que se usa en el trabajo.

AWSIoTPythonSDK.MQTTLib → Es el módulo de conexión para MQTT del SDK ofrecido por Amazon Web Services. Es necesario para la conexión por el protocolo MQTT con el IoT Core.

4.3.2.2 SQL

Se ha utilizado en este trabajo para manejar las bases de datos de MySQL. Sus siglas significan *Lenguaje de Consulta Estructurada*. Se usa en la herramienta MySQLWorkbench, en el script de Python para subir datos a MySQL y en el IoT Core cuando realizamos *acciones* por la recepción de un mensaje.

4.3.2.3 Shell

Shell es un intérprete de líneas de comando que nos permite ejecutar programas en el sistema operativo. En el trabajo se usa para ejecutar programas de Python, así como funciones específicas como crear un directorio o borrar un fichero.

4.3.3 MySQLWorkbench

Es una herramienta para la configuración y edición de bases de datos MySQL. Nos permite, sin tener un conocimiento avanzado en lenguaje SQL, cambiar la configuración de tablas en las bases de datos, realizar consultas e insertar y borrar datos, entre otras cosas. Es la herramienta recomendada cuando trabajas con MySQL.

4.3.4 Unix Cron

Es un administrador regular de procesos en segundo plano. Nos permite ejecutar procesos o guiones de manera automatizada cada cierto intervalo de tiempo o al iniciar el equipo. Los procesos se indican en el fichero crontab. En este trabajo se usa para ejecutar programas al iniciar la Raspberry o cada cierto tiempo.

4.3.5 VNC Viewer

Es una herramienta que sirve para controlar remotamente un equipo haciendo uso del protocolo TCP/IP. Es muy útil para controlar la Raspberry desde un ordenador. Para usar esta herramienta debemos seguir el tutorial publicado en la página `https://www.raspberrypi.org/documentation/remote-access/vnc/`

4.3.6 MatLab

Se ha utilizado para realizar gráficas a partir de los datos de consumo. Puede servir para muchas cosas más dentro de este campo ya que cuenta con herramientas muy avanzadas. Por ejemplo, los algoritmos de Machine Learning pueden aplicarse a los datos de consumo para realizar predicciones de consumo o identificar patrones.

4.3.7 Protocolos de comunicación

4.3.7.1 Wi-Fi

Gracias a que es una tecnología que se encuentra en la mayoría de viviendas, podemos utilizarla para la conexión a Internet de la Raspberry (como base para el envío de datos) y de los ordenadores que se usen para la visualización de datos y desarrollo del trabajo.

4.3.7.2 MQTT

Es el protocolo utilizado para el envío de datos a las tablas NoSQL. La plataforma IoT Core soporta este protocolo y HTTP.

4.3.7.3 HTTP

Se usa para realizar las consultas a las bases de datos SQL. La librería *pymysql* se basa en HTTP para ejecutar las consultas. También es posible interactuar con el IoT Core por medio de HTTP.

5 Diseño del prototipo

5.1 Requisitos

El prototipo debe contar con unos requisitos mínimos:

1. **Medir cada cierto intervalo de tiempo el consumo de la vivienda**
2. **Subir los datos del consumo a la nube**
3. **Almacenar estos datos para el análisis y tratamiento posterior**
4. **Notificar al usuario ante eventos importantes sobre el consumo**

5.2 Arquitectura de capas

El prototipo ideado es un sistema de arquitectura multicapa de hardware-software. Como hemos comentado previamente, haremos uso de servicios en la nube, pero el prototipo dispondrá de una arquitectura en la cual se tienen en cuenta más capas.

La siguiente figura muestra las capas que ocupa este prototipo y que puede servir como ejemplo para cualquier sistema del Internet de las Cosas:

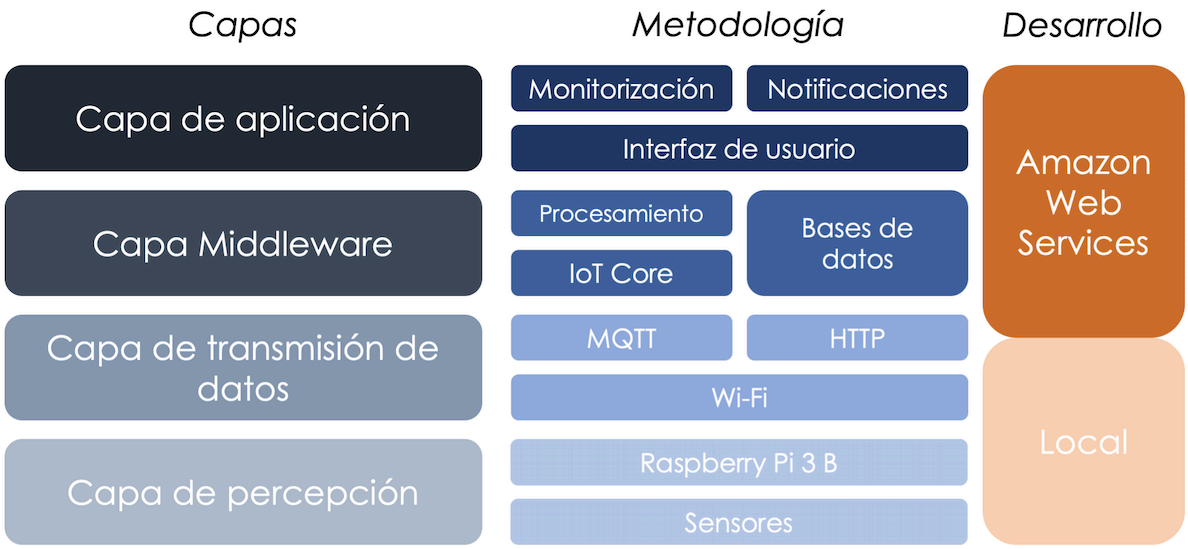


Figura 5.1: Arquitectura multicapa del prototipo.

Las capas son:

Capa de aplicación: Es la capa que ofrece AWS o cualquier plataforma que permita al usuario interactuar con el sistema, esto incluye visualizar los datos, recibir notificaciones, indicar acciones, etc.

Capa Middleware: Permite conectar servidores, procesar datos, realizar análisis, guardar elementos en bases de datos, etc. Conecta aplicaciones de distinto tipo, como por ejemplo el IoT Core y el servicio SNS de notificaciones de AWS.

Capa de transmisión de datos: Hace referencia a las tecnologías de comunicación y los protocolos que se utilizan para transmitir datos. Esto es desde los puertos de acceso a Internet hasta los protocolos como MQTT.

Capa de percepción: Esta capa incluye los dispositivos físicos que en estas aplicaciones son llamados dispositivos IoT. Esto es desde los sistemas embebidos que utilizan tecnologías de comunicación hasta los sensores o actuadores que miden (perciben) los datos del entorno.

5.3 Estructura

El siguiente diagrama muestra las conexiones del prototipo a través de las herramientas utilizadas, es decir, define la estructura de este:

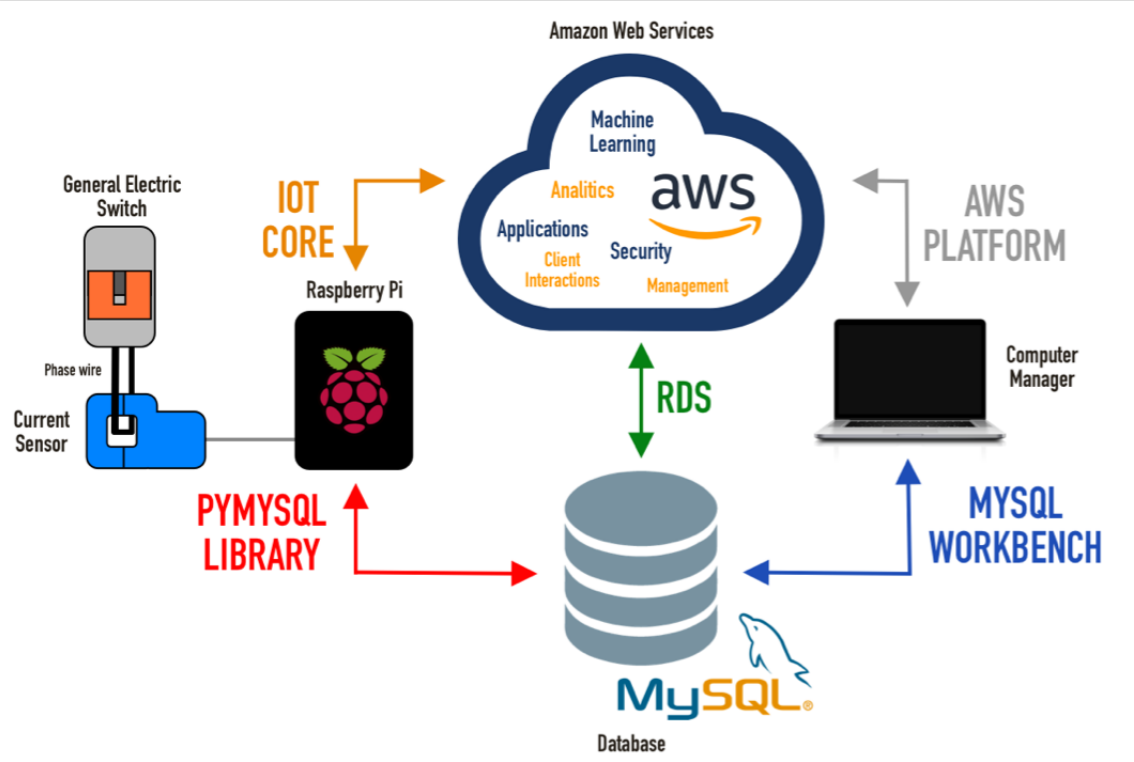


Figura 5.2: Estructura del prototipo.

5.4 Descripción

El prototipo trata de un sistema de gestión energética basado en tecnologías IoT. Dicho sistema se encarga de medir el consumo de mi vivienda mediante el uso de un sistema (que simula un sistema embebido) compuesto por una Raspberry Pi 3 B y un sensor de corriente no invasivo. Se hace uso de tecnologías de comunicación (Wi-Fi) y protocolos de comunicación (HTTP y MQTT) para el envío de datos de consumo a bases de datos MySQL y DynamoDB.

El sensor mide la corriente del cable de fase del ICP (Interruptor de Control de Potencia) de la vivienda y envía datos por cable a los pines del convertidor analógico-digital conectado sobre la Raspberry.

La conexión de la Raspberry con la plataforma IoT AWS se realiza a través del IoT Core por medio de los protocolos HTTP y MQTT.

El envío de datos de consumo a MySQL se realiza por HTTP gracias a la librería pymysql de Python.

El control de la base de datos en cuanto a configuración de parámetros de esta, creación de la base de datos, facturación y otros aspectos administrativos se realiza desde la plataforma AWS por medio de RDS (Relational Database Service).

La edición y configuración de la base de datos MySQL (creación de tablas, inserción y borrado de elementos, borrado de tablas, consultas, etc.) se realiza a través de la herramienta MySQLWorkbench instalada en el ordenador de trabajo.

La plataforma de Amazon Web Services nos ofrece una gran cantidad de herramientas para el tratamiento de los datos obtenidos. A esta plataforma se accede a través del ordenador de trabajo directamente.

Aunque no aparezca en el diagrama de la figura 5.1. VNC Viewer nos permite acceder a la Raspberry desde el ordenador de trabajo.

5.5 Presupuesto

5.5.1 Inversión inicial

Raspberry Pi →35€

Sensor de corriente →60€

Convertidor ADC →20€

Otros (Tarjeta SD, cables, etc) →30€

TOTAL →145€

Como se ha comentado anteriormente, este es un prototipo, y como en cualquier prototipo los gastos de inversión por sistema son mayores que en una producción en serie. El coste de la inversión de un sistema para la gestión energética a nivel doméstico, de cara a un negocio, podría no superar los 60€.

5.5.2 Costes variables

Los costes variables vienen principalmente de Amazon Web Services. Se tienen en cuenta los costes que puede acarrear un prototipo. A nivel comercial habría que estudiar costes extra como mantenimiento, licencias, etc.

Los costes del uso de la plataforma pueden variar mucho en función de los parámetros:

1. Qué herramientas se usan.
2. Cuánto se usan estas herramientas.
3. Almacenamiento en las bases de datos.
4. Conexiones con la plataforma.

5.5.2.1 IoT Core

Para un prototipo compuesto por 1 dispositivo que realiza 1 actualización de sombra por segundo y en el cual se insertan datos en una base de datos NoSQL de DynamoDB, enviando una media de 24 mensajes al día al usuario:

Conectividad →0,0031€

Mensajería →0,004€

Actualización de sombras →6,48€

Acciones por reglas →0,4€

TOTAL →6,88€ al mes / 82,56€ al año

Para más información sobre los precios consultar `https://aws.amazon.com/es/iot-core/pricing/`

5.5.2.2 MySQL

Las transferencias entrantes de datos en MySQL son gratuitas y las salientes hasta 1GB al mes también. El tipo de base de datos utilizada puede ser la más económica (db.t3.micro), ya que se ha probado y funciona perfectamente para este prototipo. Por último, se guardan 82MB al mes de información. Esto supone:

Instancia reservada →6,48€

Almacenamiento →0,01€

TOTAL →6,5€ al mes / 78€ al año

Para más información sobre los precios consultar `https://aws.amazon.com/es/rds/mysql/pricing/`

5.5.2.3 DynamoDB

Cada día el sistema guarda en la base de datos NoSQL 86400 elementos (cada segundo, una lectura). Esto se traduce en 5,5MB de almacenamiento diarios, que al mes son 165MB. Puesto que en DynamoDB los primeros 25GB al mes son gratuitos, el almacenamiento es gratuito para este prototipo. Las escrituras son 86400 al día y las lecturas variarán en función de las herramientas utilizadas, supongamos las mismas lecturas que escrituras para realizar un análisis de predicción de consumo.

Escritura →3,66€

Lectura →0,648€

TOTAL →4,31€ al mes / 51,7€ al año

Para más información sobre los precios consultar `https://aws.amazon.com/es/dynamodb/pricing/`

5.5.2.4 Otras herramientas

Amazon Web Services cuenta con herramientas de análisis, procesamiento de datos, Machine Learning, Deep Learning, servicios de mensajería, etc. Según la aplicación y el uso de estas herramientas, como hemos comentado, los costes mensuales pueden variar en gran medida.

5.5.2.5 Consumo eléctrico

El consumo del sistema básicamente es el consumo de la Raspberry Pi. Esta tiene una potencia de 1,8W a pleno rendimiento. Al mes se traduce en 73Wh y al año en 876Wh. Esto supone un gasto de aproximadamente 10 céntimos de euro al año.

5.5.3 Coste total

La inversión inicial en este prototipo es de aproximadamente 145€. El coste básico mensual aproximado de este prototipo es de 12,28€ al mes/147,42€ al año.

5.5.4 Ahorro

Para una familia que paga 1000€ al año en la factura de la luz, necesitamos que el sistema genere un 25% de ahorro energético para ser rentable en 4 años, con los costes de inversión y variables anteriormente comentados.

6 Desarrollo

6.1 Instalación del equipo y conexiones

Como hemos visto en el diagrama de la figura 5.1. el equipo se ha colocado próximo al cuadro eléctrico de la vivienda. El sensor de corriente se ha instalado, como se ha comentado previamente, en el ICP.

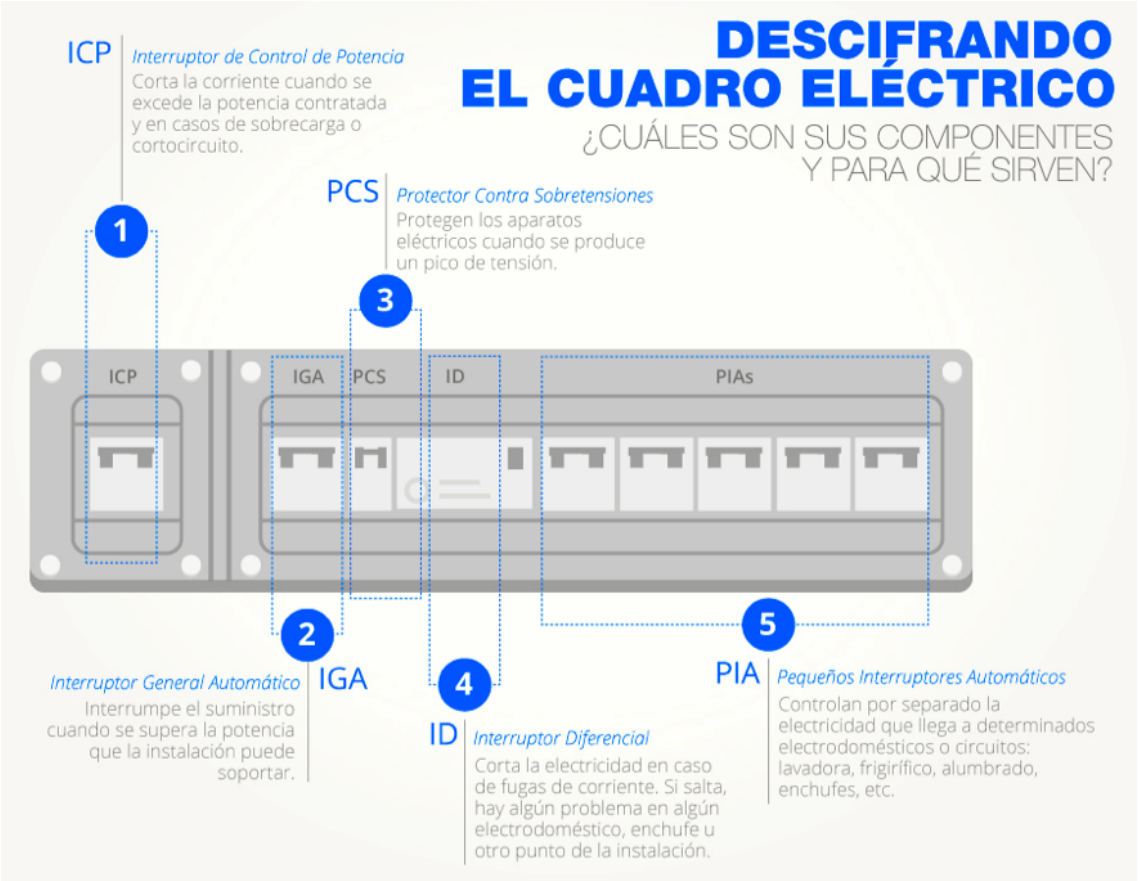


Figura 6.1: Componentes del cuadro eléctrico de una vivienda. Fuente: Endesa

Figura 6.2: Instalación del sensor de corriente AT100 B10 en el ICP.

La conexión del sensor con el convertidor ADC se realiza mediante cables. La salida positiva del sensor se conecta al pin A0 y la negativa al pin GND. El convertidor se coloca encima de la Raspberry como se ha indicado en la figura 4.4. Finalmente, la Raspberry se conecta a la corriente de un enchufe situado próximo al cuadro eléctrico.

Una vez instalado el equipo, la conexión con la Raspberry se puede realizar mediante cable Ethernet o Wi-Fi. El ordenador de trabajo se encuentra conectado a la red y por medio de la herramienta VNC Viewer nos conectamos a la Raspberry.

Figura 6.3: Instalación del prototipo.

Para obtener la dirección IP de la Raspberry hay que hacer uso del comando en la terminal:

Código 6.1: Comando para obtener la dirección IP en la Raspberry

```
1        pi@raspberrypi:~ $ ifconfig
```

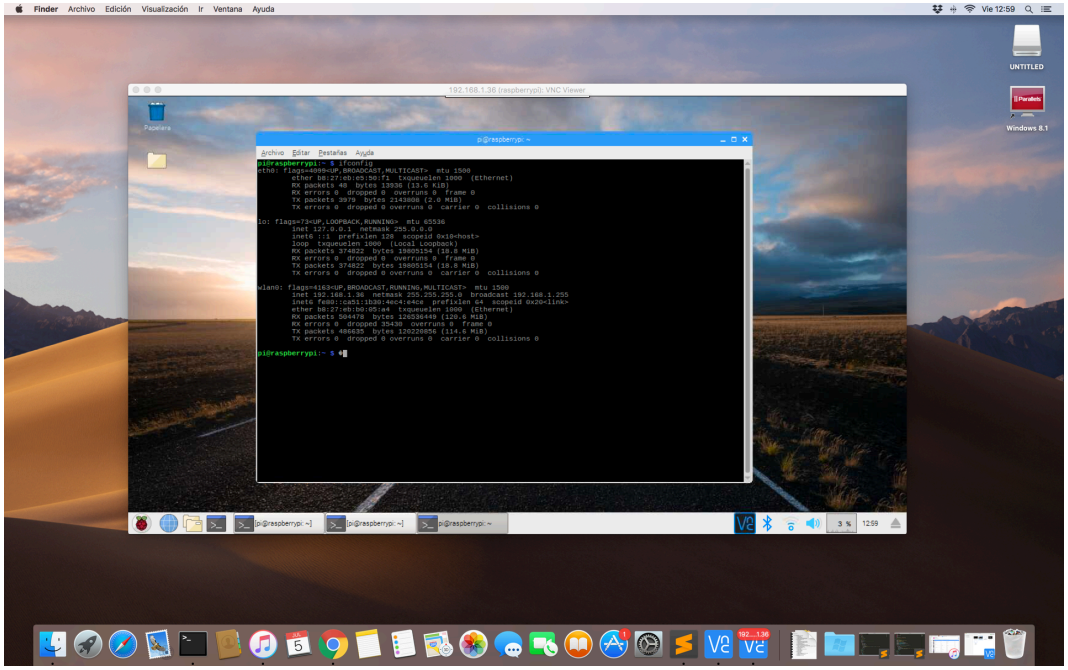


Figura 6.4: Visualización del escritorio de la Raspberry desde el ordenador de trabajo.

Vemos en la figura 6.4. que la dirección IP en nuestra Raspberry es *192.168.1.36*. Ahora ya tenemos acceso a la Raspberry remotamente.

Importante: La dirección IP de la Raspberry es dinámica, por lo que puede cambiar cada vez que se reinicia. La dirección IP por cable Ethernet es estática.

6.2 Instalación del software

Una vez tenemos los equipos instalados y conectados, se procede a la instalación de las aplicaciones y herramientas necesarias para el sistema. Estas son:

1. MySQLWorkbench
2. VNC Viewer
3. MatLab
4. Librerías
   a) SunFounder_PiPlus.Shield
   b) pymysql
   c) AWSIoTPythonSDK.MQTTLib

6.3 Calibrado del sensor

La calibración de sensores es un aspecto muy importante en aplicaciones de este tipo. En algunos sensores se puede realizar una calibración directamente en el sensor. En nuestro caso la calibración tenemos que realizarla por software. Este proceso es relativamente sencillo. Se ha usado un coeficiente que se ha obtenido de forma empírica. Es decir, se han encendido distintos electrodomésticos de potencia conocida y se ha hecho una regla de tres para obtener la potencia consumida aproximada.

El coeficiente que se ha obtenido es: **44,23**.

Este valor se multiplica por la medida obtenida por el sensor, dando una potencia aproximada a la real. Sabiendo que la tensión de la red eléctrica es constante (teóricamente), que:

$$P = I \cdot V \tag{6.1}$$

Siendo P: Potencia, I: Corriente y V: Tensión

Y que la potencia (y por tanto el consumo) es directamente proporcional a la corriente, midiendo la corriente y aplicando el coeficiente podemos obtener el consumo aproximado de la vivienda.

Para usar el ADC y obtener las lecturas del sensor, según el tutorial ofrecido por el fabricante y mencionado en el apartado 4.2.3., debemos usar Python. Declaramos una instancia del ADC y leemos la entrada del pin A0 de la siguiente manera:

Código 6.2: Leer del pin A0 del ADC

```
1        ADC = PCF8591()
2        data = ADC.read(0) #Lectura de la entrada A0
```

6.4 Creación de bases de datos

Diferenciamos entre las bases de datos SQL y NoSQL:

6.4.1 MySQL

Para crear una base de datos en MySQL debemos seguir los siguientes pasos:

1. Entrar en la plataforma Amazon Web Services
2. Acceder al servicio de RDS (Relational Database Services)
3. En el apartado *Databases*, seleccionar *Create database*
4. Seleccionar el proveedor MySQL
5. Seleccionar uso de desarrollador o pruebas (*Dev/Test-MySQL*)

6. Elegir el tipo de base de datos (*db.t2.micro*)
7. Tendremos en cuenta el uso de la capa gratuita de AWS seleccionando la opción *Only enable options eligible for RDS Free Usage Tier*
8. Asignaremos nombre, usuario y contraseña
9. Asignaremos el grupo de parámetros (Es importante crear un grupo de parámetros que no sea el asignado por defecto para poder editar los parámetros)

El grupo de parámetros se crea de la siguiente forma:

1. Entrar en la plataforma Amazon Web Services
2. Acceder al servicio de RDS (Relational Database Services)
3. En el apartado *Parameter groups*, seleccionar *Create parameter group*
4. Seleccionar la familia coincidente con la versión de la base de datos creada (*mysql5.7*)
5. Asignar nombre y descripción

Una vez creamos el grupo de parámetros, podemos seleccionar nuestra base de datos en el apartado *Databases* y asignarle el nuevo grupo. Este grupo lo podemos modificar, por ejemplo, para cambiar la zona horaria. Esto sirve para añadir valores de tiempo por defecto a la base de datos que se correspondan con la hora local.

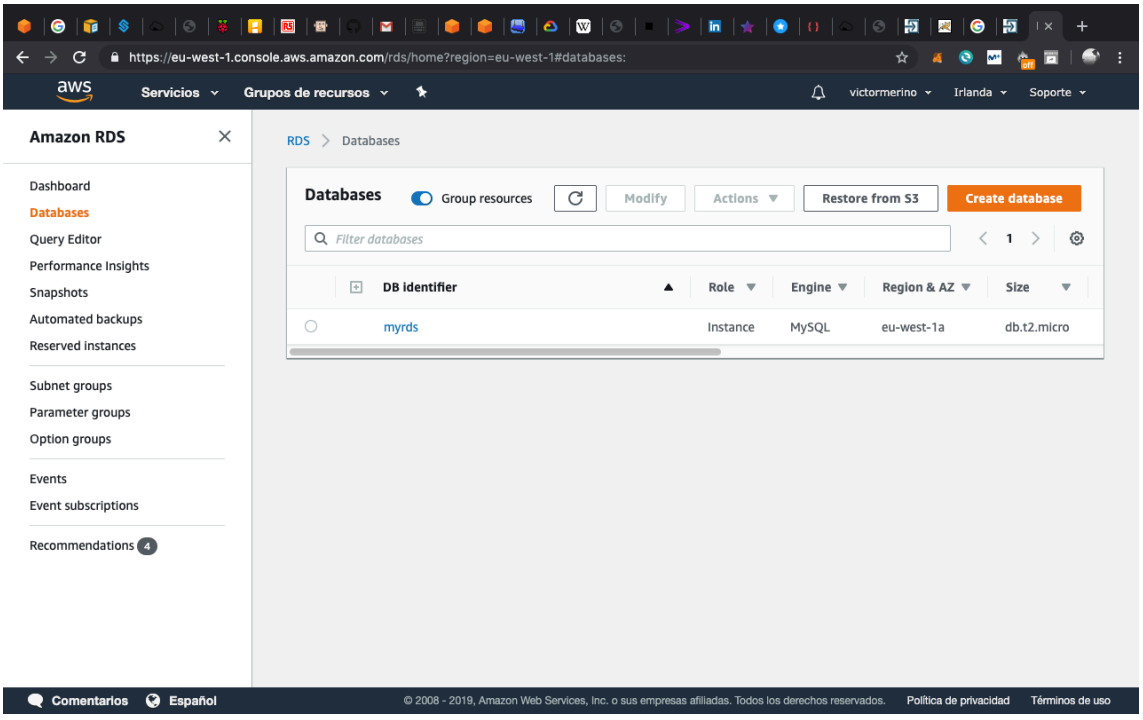


Figura 6.5: Pantalla de RDS con la base de datos creada.

Para crear tablas dentro de la base de datos utilizamos la herramienta MySQLWorkbench. El proceso es el siguiente:

1. Conexión a través del endpoint (en las propiedades de la base de datos, es del tipo *myrds.xxxxxxx.eu-west-1.rds.amazonaws.com*), del nombre y la contraseña

2. En el panel de la izquierda hacemos clic derecho en *Tables* y clic derecho en *Create table...*

3. Creamos dos columnas (*timestamp* y *w*)

4. La columna del tiempo puede ser tipo *TIMESTAMP* o *INT* (Si asignamos *TIMESTAMP* podemos asignar un valor por defecto para el tiempo de inserción de la hora local, por eso se cambió el grupo de parámetros)

5. La columna del consumo *w* puede ser de tipo *DOUBLE* o *INT*

6. La *Primary Key* es la columna que nunca puede estar vacía y que sirve para ordenar los elementos, seleccionamos *timestamp*

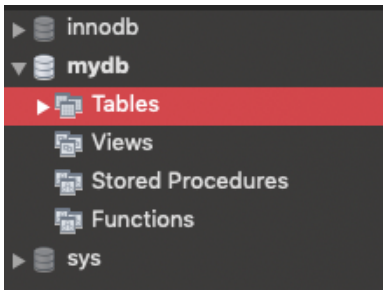


Figura 6.6: Esquema de la base de datos MySQL desde MySQLWorkbench.

En MySQL Workbench se pueden ejecutar comandos SQL desde las *Query Tab*. Todas las acciones de configuración son comandos SQL que se envían a la base de datos, pero no necesitamos saberlos para configurarla gracias a esta herramienta.

Figura 6.7: Ventana de configuración de la tabla en MySQLWorkbench.

6.4.2 DynamoDB

La creación de bases de datos NoSQL es más sencilla:

1. Entrar en la plataforma Amazon Web Services
2. Acceder al servicio de DynamoDB
3. En el apartado *Tablas*, seleccionar *Crear tabla*
4. Asignamos un nombre a la tabla (*IoTSystem*)
5. Asignamos una clave de partición (*IoTObject*) y su tipo (*string*)
6. Asignamos una clave de ordenación (*Timestamp*) y su tipo (*number*)

La tabla se crea como si fuera para una aplicación con más dispositivos, por eso se asigna una clave de partición que identificará los dispositivos y una clave de ordenación que ordenará los datos de cada dispositivo por tiempo.

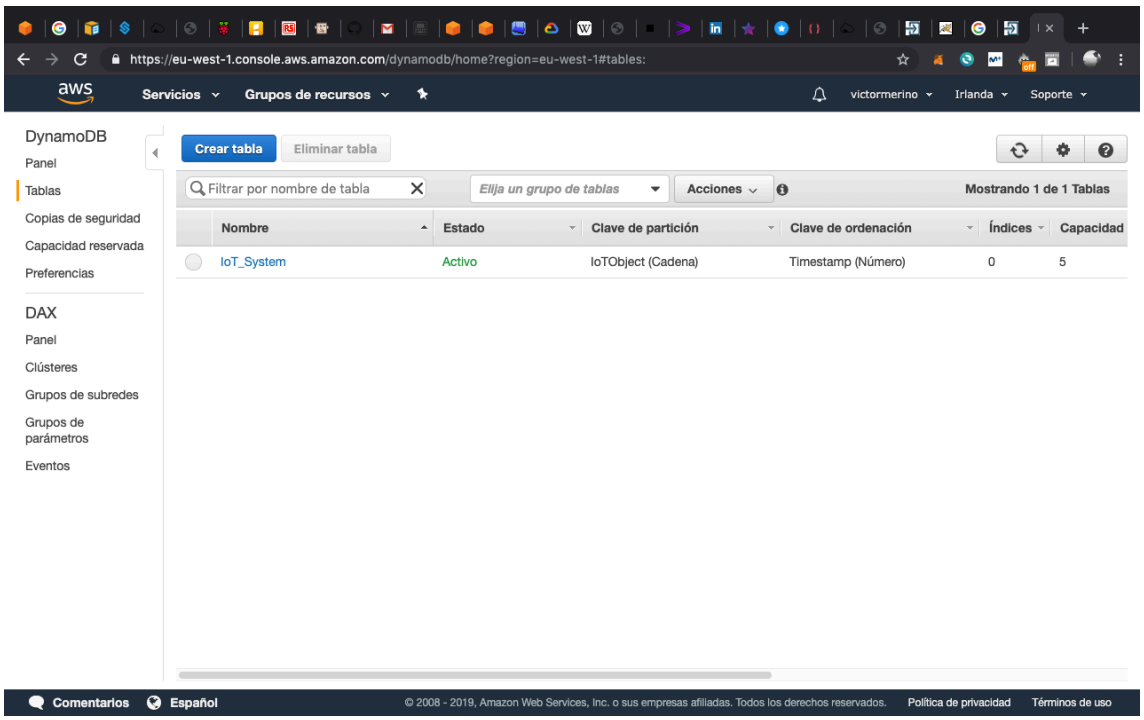


Figura 6.8: Pantalla de DynamoDB con la base de datos creada.

6.5 IoT Core

El proceso seguido para el uso del IoT Core para el trabajo ha sido el siguiente:

6.5.1 Creación de una política

El primer paso es crear una política para asignar esta al dispositivo. Esta política dota al objeto de distintos permisos dentro de la plataforma IoT Core. No debemos confundirlas con las políticas de la plataforma AWS, que dota de permisos a ciertos roles dentro de AWS.

1. Accedemos al servicio IoT Core de AWS
2. En la sección *Seguro* →*Políticas* haremos clic en *Crear*
3. Añadimos nombre a la política (*rpi-policy*)
4. Asignamos todas las acciones (*iot:**, permitiremos que realice todas)
5. Asignamos el ARN del recurso (*, para seleccionar cualquiera)
6. Seleccionamos permitir y creamos la política

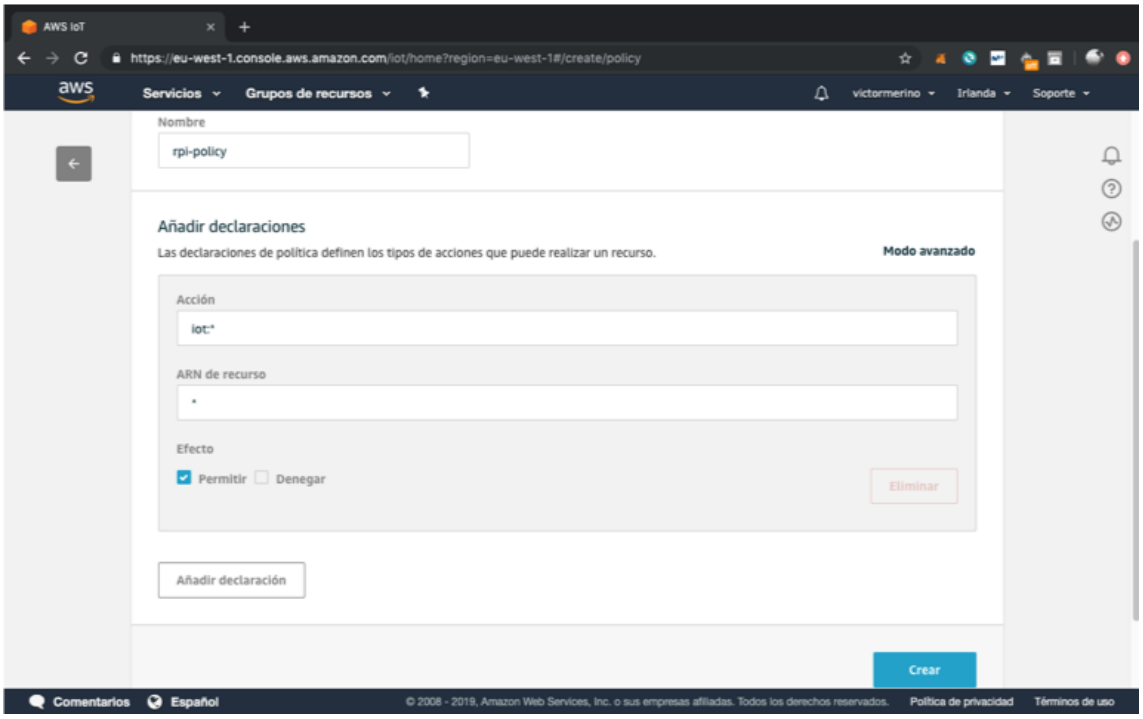


Figura 6.9: Pantalla de creación de una política en IoT Core.

6.5.2 Creación de un objeto

La creación de un objeto sigue los siguientes pasos:

1. Accedemos al servicio IoT Core de AWS
2. En la sección *Administración* →*Objetos* haremos clic en *Crear*
3. Hacemos clic en *Crear un solo objeto*
4. Es el momento de crear un certificado

6.5.3 Certificado y claves

La creación del certificado es importante y necesaria para garantizar la seguridad de las comunicaciones con el objeto. Se utiliza en los programas de Python que utiliza el protocolo MQTT. Así como la clave privada.

En la pantalla en la que nos encontrábamos al crear el objeto, seguiremos el proceso:

1. Hacemos clic en *Crear un certificado*
2. Descargamos el certificado y las claves pública y privada
3. Accedemos a la página externa para descargar *Amazon Root CA 1*
4. Activamos el certificado

6.5.4 Sombra y temas

Los protocolos de comunicación publisher-subscriber se basan en temas. En estos temas los publishers pueden editar el estado del tema y los subscribers pueden leer el estado del tema. Los objetos del IoT Core se basan en 10 temas que interactúan con la sombra del objeto. Esta sombra es el estado del objeto. Los temas son:

update →Actualizar la sombra del objeto

update/accepted →Se aceptó la actualización de la sombra del objeto

update/documents →Actualizar los documentos de la sombra

update/rejected →Se rechazó la actualización de la sombra del objeto

get →Obtener la sombra del objeto

get/accepted →Se aceptó la obtención de la sombra del objeto

get/rejected →Se rechazó la obtención de la sombra del objeto

delete →Eliminar la sombra del objeto

delete/accepted →Se aceptó la eliminación de la sombra del objeto

delete/rejected →Se rechazó la eliminación de la sombra del objeto

Para actualizar la sombra del objeto usamos el tema update. Si la sombra se ha actualizado, recibimos la sombra del objeto junto a los metadatos en el tema update/accepted. Si no se ha podido actualizar, recibimos un mensaje de error en el tema update/rejected.

Para obtener la sombra del objeto, debemos escribir un JSON vacío {} en el tema get. Si se acepta, recibimos la sombra del objeto y los metadatos en el tema get/accepted. Si no se acepta, recibimos un mensaje de error en el tema get/rejected. De la misma forma que el get, podemos usar el tema delete ya que funciona de manera semejante.

6.5.5 Reglas y acciones

Las acciones son herramientas que están dentro del IoT Core. A partir de un mensaje recibido en un tema, podemos definir unas reglas para generar estas acciones, en caso de cumplirse, para guardar los datos en una base de datos (SQL, NoSQL), ejecutar funciones (Lambda), enviar notificaciones (SNS), etc.

Para definir una regla seguimos el proceso:

1. Accedemos al servicio IoT Core de AWS
2. En la sección *Actuar* haremos clic en *Crear* para crear una regla
3. Añadimos nombre *IoT_to_Dynamo*

4. Añadimos instrucción de consulta de regla

5. Definimos las acciones a tomar

La instrucción de consulta de regla se escribe en lenguaje SQL. Cuando se publica un mensaje en el tema seleccionado se ejecuta la regla. Si no hay ningún error y se cumple la regla, se ejecutan las acciones definidas.

La regla para la inserción de los datos en la base de datos de DynamoDB es:

Código 6.3: Regla para la inserción de datos en DynamoDB

```
SELECT state.reported.* FROM '$aws/things/MyRPi/shadow/update/accepted'
```

Los mensajes publicados en el tema *update/accepted* son en formato JSON. Para seleccionar los datos que nos interesan (la sombra), usamos *state.reported.**.

Un ejemplo de JSON del tema *update/accepted* es:

Código 6.4: JSON de la sombra del objeto

```
"payload": {
  "state": {
    "reported": {
      "Timestamp": 1562349396,
      "IoTObject": "RaspberryPi",
      "Data": {
        "Temperature": 25,
        "Power": 1939
      }
    }
  },
  "metadata": {
    "reported": {
      "Timestamp": {
        "timestamp": 1562349397
      },
      "IoTObject": {
        "timestamp": 1562349397
      },
      "Data": {
        "Temperature": {
          "timestamp": 1562349397
        },
        "Power": {
          "timestamp": 1562349397
        }
      }
    }
  },
  "version": 333294,
  "timestamp": 1562349397,
  "clientToken": "055097a5-8aa9-4d84-933b-f3a27aad0da5"
}
```

La acción utilizada para guardar los datos en DynamoDB es *Dividir mensajes en varias columnas de una tabla de base de datos (DynamoDBv2)*. Esta acción añade los elementos del JSON de manera automática.

6.5.6 Conexión por MQTT

Para conectar el dispositivo (Raspberry) por MQTT al IoT Core se utiliza la librería *AWSIoTPythonSDK.MQTTLib*. Es necesaria la inclusión de la librería en el programa de Python que se use y, además debemos usar, los archivos descargados: certificado, clave privada y CA raíz.

El script de prueba de conexión al IoT Core en Python desde la Raspberry es:

Código 6.5: Script Python de prueba de subida a IoT Core

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTShadowClient

#Funcion que se ejecuta al actualizar la sombra (vacía)
def myShadowUpdateCallback(payload, responseStatus, token):
  return

#Funcion de subida de datos a la nube
def toIoTCore():

    #Declaracion de variables
    SHADOW_CLIENT = "shadowClient"
    HOST_NAME = "xxxxxxxx-ats.iot.eu-west-1.amazonaws.com"
    ROOT_CA = "AmazonRootCA1.pem"
    PRIVATE_KEY = "38fe8c20c0-private.pem.key"
    CERT_FILE = "38fe8c20c0-certificate.pem.crt"
    SHADOW_HANDLER = "MyRPi"
    #Configuracion del cliente
    myShadowClient = AWSIoTMQTTShadowClient(SHADOW_CLIENT)
    myShadowClient.configureEndpoint(HOST_NAME, 8883)
    myShadowClient.configureCredentials(ROOT_CA, PRIVATE_KEY,CERT_FILE)
    myShadowClient.configureConnectDisconnectTimeout(10)
    myShadowClient.configureMQTTOperationTimeout(5)
    #Conexion del cliente
    myShadowClient.connect()
    #Manejador de la sombra
    myDeviceShadow = myShadowClient.createShadowHandlerWithName(SHADOW_HANDLER, True)
    #Declaracion de un JSON
    json = '{ "state": { "reported":{ "IoTObject": "RaspberryPi","Timestamp": 1562349396, "Data":{ "Power↩
        ↪ ": 1939, "Temperature": 25}}}}'
    #Actualizacion de la sombra
    myDeviceShadow.shadowUpdate(json,myShadowUpdateCallback, 5)

toIoTCore()
```

6.6 Diseño del sistema

Como se ha comentado anteriormente, el objetivo del sistema es obtener los datos de los sensores y subir estos a la nube para analizarlos y tomar decisiones. En un primer momento, se trató de hacer un único programa que realizara el proceso de lectura del sensor y la subida de datos, pero surgen problemas a la hora de ejecutar el programa. Por lo que el diseño del sistema ha cambiado.

Mientras el programa está comunicándose con la nube, no estamos leyendo la información del sensor, es decir, estamos perdiendo información. Esto está relacionado con el problema de los *sockets bloqueantes.*

Sockets bloqueantes: La mayoría de las comunicaciones con la nube se basan en sockets que cuando no hay conexión se bloquean, deteniendo el programa en ese punto hasta que se recupere la conexión, por lo que se pierde información.

Un socket es un concepto abstracto. Se considera una conexión establecida entre dos puntos que garantiza el flujo de datos de manera fiable y ordenada. Cuando usamos TCP, la mayoría de las implementaciones de sockets establecen sockets bloqueantes, lo que puede ser un problema para aplicaciones como esta.

6.6.1 Guardado de datos en local

La idea de guardar los datos en local es que no se produzca una pérdida de datos si la conexión a Internet cae o si se produce un error con la nube. Cuando se recupera la conexión a la nube, los datos en local se suben a la nube y se borran del almacenamiento local, ya que la Raspberry cuenta con una capacidad de almacenamiento limitada.

Como solución a la pérdida de datos por los problemas anteriores se ha propuesto una solución efectiva que es separar el funcionamiento en dos programas.

Programa 1: Guarda los datos en local continuamente.

Programa 2: Sube los datos a la nube cuando haya conexión y borra los datos subidos de la memoria local. Si el programa se bloquea por falta de conexión, el Programa 1 sigue leyendo datos de consumo.

Se han explorado distintas formas de guardar los datos en local:

1. Un archivo CSV. Aunque *a priori* pueda parecer la opción más cómoda:

   Es una opción lenta: Abrir archivo, escribir los datos, cerrar el archivo, volverlo a abrir para leer datos, borrar datos… Todas estas acciones se deben realizar en un corto período de tiempo.

   Es difícil manejar un CSV en tiempo real: Como debemos borrar los datos, es complicado borrar un dato en un archivo CSV y recolocar todos los datos en el archivo en tiempo real.

2. Un archivo CSV por segundo con el tiempo y el consumo. Opción más rápida ya que solo tenemos que leer el archivo y borrarlo.

3. Archivo vacío. Guardamos la información en el nombre del archivo:

   Es la opción más rápida: No necesitamos usar un lector de archivo, simplemente conociendo el nombre del archivo obtenemos la información y borramos el archivo.

   Menos errores: No se producen errores al borrar archivos mientras se está leyendo o escribiendo en él.

Esta última opción es la que se está utilizando en el prototipo. Contamos con un directorio llamado *noconn_db* en el que se guardan los archivos de las lecturas como podemos ver en la figura siguiente:

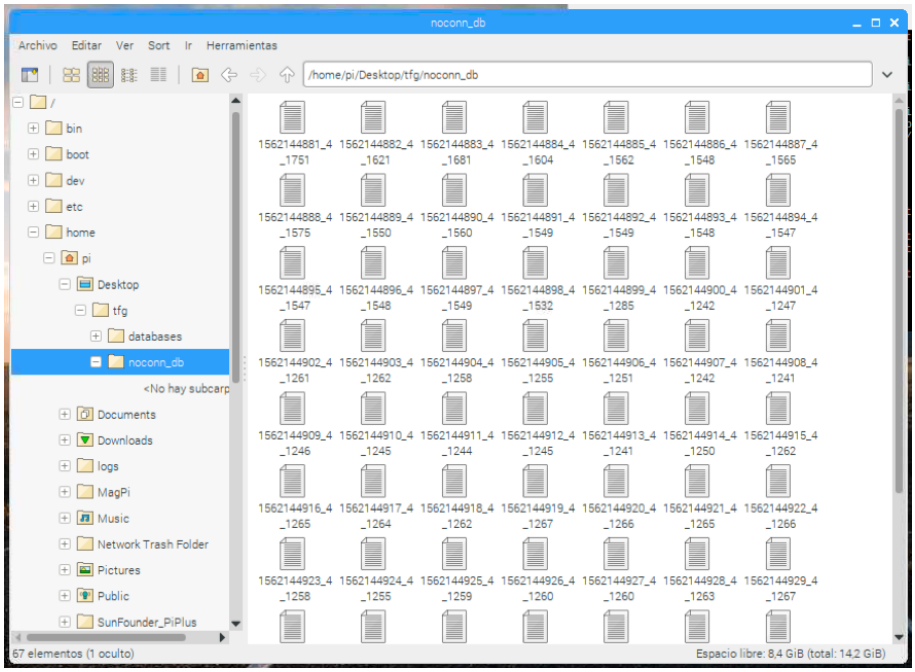


Figura 6.10: Directorio de datos en local en la Raspberry.

El formato es Timestamp_CifrasDeConsumo_Consumo. Se utiliza este formato para facilitar la lectura del archivo, ya que la cifra del consumo varía entre 3 y 4 cifras y a la hora de leer el archivo puede ocasionar errores; por lo que se indica el número de cifras previamente, que funciona para cualquier cifra de potencia desde 0 a 10 millones de kW.

6.6.2 Automatización de procesos

La automatización de procesos hace referencia a la ejecución de los programas del sistema de manera automática cuando se inicia la Raspberry. De esta manera no es necesaria la ejecución remota de los programas si reinstalamos el prototipo en otro lugar o si se produce un apagón.

Para conseguir esta funcionalidad se ha usado la herramienta mencionada en el apartado 4.3.4. Para instalar Cron en la Raspberry y poder añadir comandos a la tabla Cron usamos:

Código 6.6: Instalación Unix Cron y programación de comandos

```
1 sudo apt-get install gnome-schedule
2 crontab -e
```

El formato del comando para añadir una tarea es el siguiente:

#m h dom mon dow command

Siendo m: minuto, h: hora, dom: día del mes, mon: mes, dow: día de la semana y command: tarea a ejecutar. También podemos añadir *@reboot* antes del comando para ejecutar este al iniciar la Raspberry, justo lo que queremos. Se han añadido estas líneas al archivo:

Código 6.7: Comandos añadidos en Crontab

```
1 @reboot sh /home/pi/launcher.sh > /home/pi/logs/cronlog 2>&1
2 #0 * * * * sh /home/pi/launcher.sh > /home/pi/logs/cronlog 2>&1
```

La primera línea es la que se ejecuta (no está comentada, como la segunda). Se ejecuta al iniciar la Raspberry. El comando del final fue necesario añadirlo para un error que ocurría a la hora de ejecutar. La segunda línea comentada se usa para ejecutar el comando cada hora en el minuto 0.

6.6.3 Tiempo de muestreo y subida

El tiempo de muestreo es un aspecto fundamental cuando tenemos una aplicación con sensores. En este prototipo se realiza una lectura asíncrona del sensor, contando las lecturas realizadas y haciendo una media de estas al final de cada segundo. Se llegan a hacer más de 1000 lecturas por segundo.

Cuando utilizamos la nube, tenemos que tener en cuenta que cuanto más espacio se ocupe y más datos subamos por minuto, más costará nuestra aplicación. Por eso no suelen enviarse datos cada milisegundo. Se suelen utilizar los milisegundos para aplicaciones en local y de segundos o más para la nube. Los mayores problemas que nos podemos encontrar son:

Almacenamiento: En este trabajo se han llegado a registrar más de 100 millones de lecturas por día, ocupando más de 2 Gigabytes en el almacenamiento de la Raspberry.

Capacidad de escritura en la nube: En plataformas como el IoT Core de AWS existen límites de escritura que si se superan puede fallar la aplicación.

En este prototipo se leen datos con precisión del milisegundo, pero se suben datos con precisión del segundo. Principalmente se usa el segundo ya que se almacenan los datos y no debemos ocupar un espacio excesivamente grande. Incluso se ha experimentado con subir datos cada minuto, pero la información perdida es grande, ya que en un minuto pueden ocurrir muchos eventos, como el uso de un secador y que se supere la potencia máxima, por ejemplo.

6.6.4 Diagrama de flujo principal

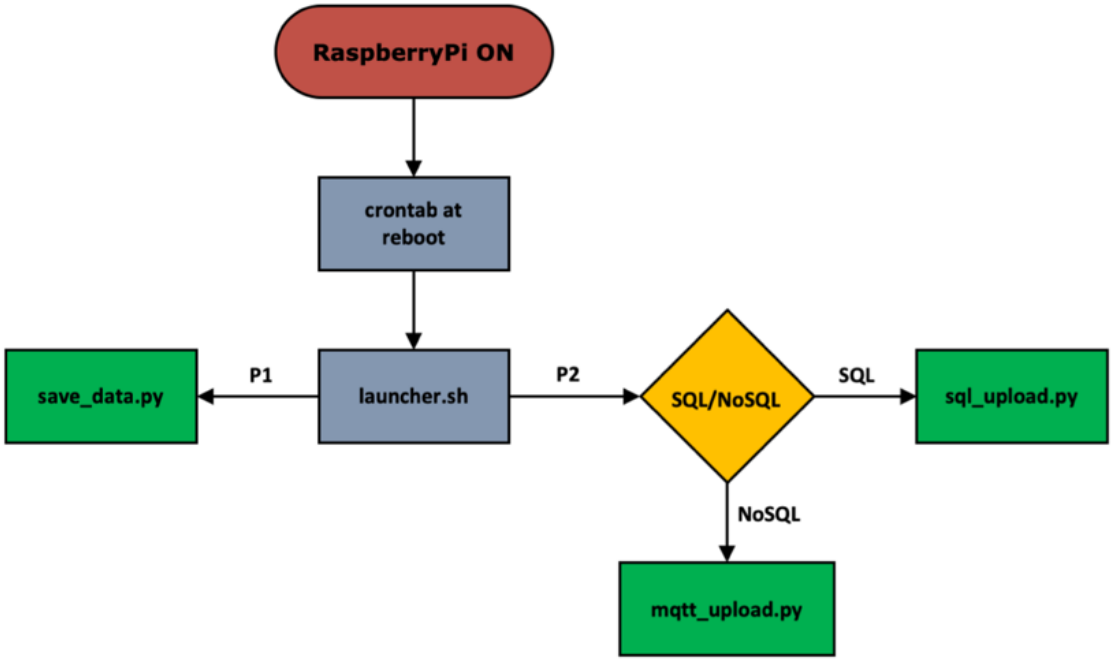


Figura 6.11: Diagrama de flujo principal del sistema.

1. Se inicia la Raspberry

2. Crontab ejecuta launcher.sh

3. Launcher.sh ejecuta P1 y P2

   P1 →Guarda los datos en local

   P2 →Cambiamos launcher.sh si queremos subir datos a una base de datos SQL o NoSQL. Si queremos los datos en SQL ejecutamos sql_upload.py, si queremos NoSQL ejecutamos mqtt_upload.py.

6.6.5 Diagrama de flujo del programa de guardado de datos (P1)

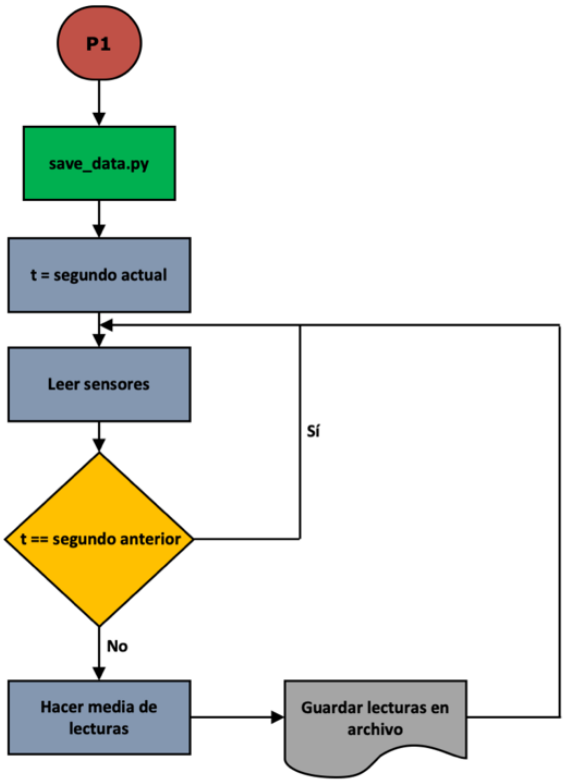


Figura 6.12: Diagrama de flujo del programa 1.

El primer programa (P1) corresponde al script de Python save_data.py cuyo código ha sido incluido en el apartado de programación de este trabajo. El algoritmo funciona de la siguiente manera:

1. Inicia el programa
2. Se guarda el tiempo actual en t
3. Lectura del sensor
4. ¿Es el mismo segundo que el de la iteración anterior?

   Sí →Sumamos el valor a una variable y volvemos al punto 3.

   No →Hacemos la media de todos los valores registrados en el segundo y guardamos el valor en local. Reiniciamos variables y volvemos al punto 3.

6.6.6 Diagrama de flujo del programa de subida de datos a MySQL (P2)

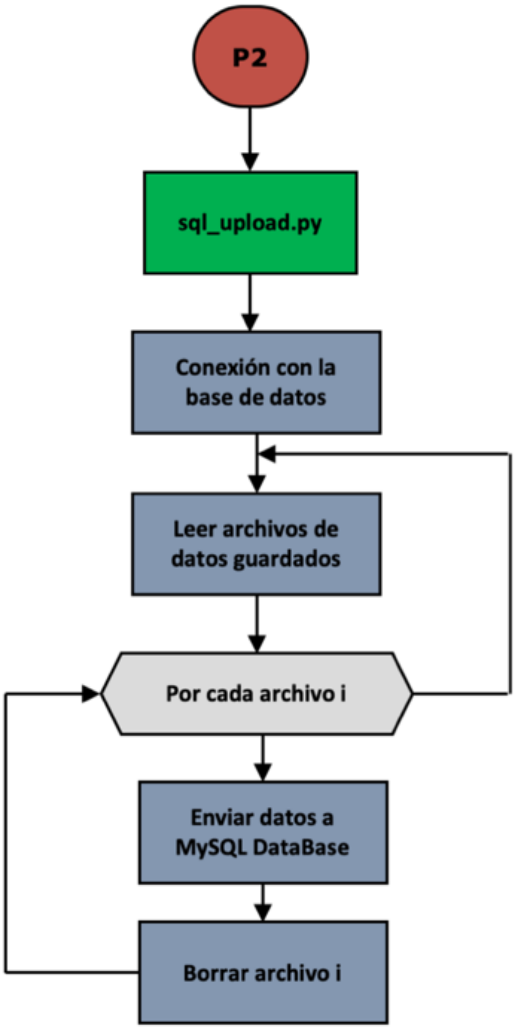


Figura 6.13: Diagrama de flujo del programa 1 para SQL.

1. Inicia el programa

2. Se establece la conexión con la base de datos

3. Leemos el directorio para conseguir los datos guardados

4. Por cada archivo en el directorio →Enviamos datos a MySQL y borramos el archivo

5. Si no hay archivos leemos el directorio continuamente

6.6.7 Diagrama de flujo del programa de subida de datos a DynamoDB (P2)

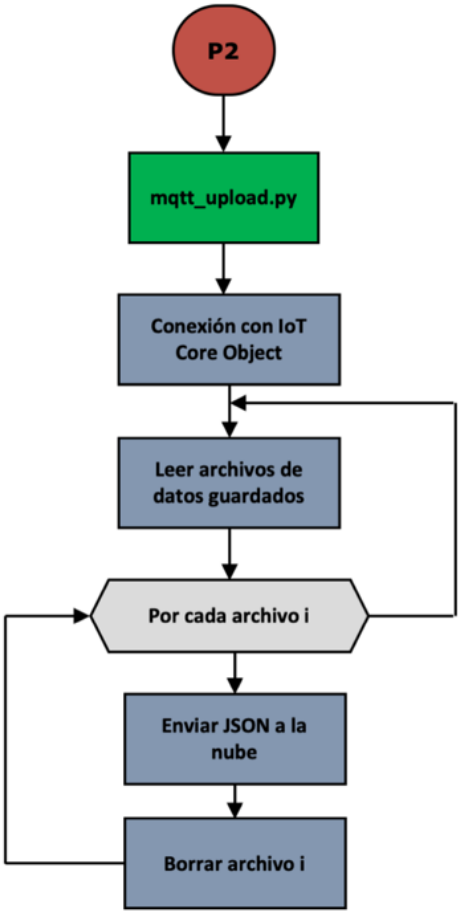


Figura 6.14: Diagrama de flujo del programa 2 para NoSQL.

1. Inicia el programa

2. Se establece la conexión con el IoT Core

3. Leemos el directorio para conseguir los datos guardados

4. Por cada archivo en el directorio →Generamos un JSON, enviamos datos por MQTT al IoT Core y borramos el archivo

5. Si no hay archivos leemos el directorio continuamente

6.7 Programación

6.7.1 launcher.sh

Función: Ejecutar los programas de Python save_data.py y sql_upload.py/mqtt_upload.py. Se lanza desde Cron directamente al iniciar la Raspberry.

Código 6.8: Script de launcher.sh

```
cd /
python /home/pi/Desktop/tfg/save_data.py
python /home/pi/Desktop/tfg/mqtt_upload.py
```

6.7.2 save_data.py

Función: Leer los datos de consumo del sensor de corriente y guardar estos en la memoria de la Raspberry.

Código 6.9: Script de guardado de datos de consumo en local

```
from SunFounder_PiPlus.Shield import *
import sys
import time

#Funcion de iniciar el ADC
def setup():
    global ADC
    ADC = PCF8591()
#Funcion de lectura del pin
def read(channel):
    return ADC.read(channel)
#Funcion de lectura continua de datos de consumo
def continous_reading():
    #Declaracion de variables
    coef = 44.23 #Coeficiente obtenido
    t = 0 #segundos
    start = time.time() #inicial
    current = time.time() #utilizamos current para guardar hasta la fecha final
    aux_cont = 0 #contador de lecturas por segundo
    aux_cons = 0 #consumo acumulado
    aux_ant = 0 #segundo anterior
    #Bucle infinito
    while True:
        value = (read(0)*coef) #valor de potencia/consumo
        current = time.time()
        t = current-start
        aux_cons = aux_cons + value
        aux_cont = aux_cont + 1

        #Si cambiamos de segundo
        if int(t) != aux_ant:
            #Hacemos la media de la potencia en ese segundo
            aux_cons = aux_cons/aux_cont
            w = str(int(aux_cons)) #Vatios
            #Guardamos archivo con el segundo anterior, las cifras del consumo y el consumo
            with open('/home/pi/Desktop/tfg/noconn_db/'+str(int(current-1))+'_'+str(len(w))+'_'+w,'w') as newfile:
                newfile.close()
            aux_cont = 0
```

```
39              aux_cons = 0
40
41          aux_ant = int(t)
42
43 def main():
44     continous_reading()
45
46 def destroy():
47     ADC.destroy()
48     GPIO.cleanup()
49
50 if __name__ == "__main__":
51
52     time.sleep(1)
53     try:
54         setup()
55         main()
56     except KeyboardInterrupt:
57         destroy()
```

6.7.3 sql_upload.py

Función: Subir los datos de consumo a una base de datos MySQL (SQL), cuando haya conexión a Internet.

Código 6.10: Script de subida de datos de consumo a MySQL

```
1 import time
2 import pymysql
3 import os
4 import glob2
5
6 #Funcion de subida de datos continua a MySQL
7 def upload():
8
9     #Declaracion de variables
10    host = "myrds.xxxxxxxx.eu-west-1.rds.amazonaws.com"
11    name = "master"
12    password = "xxxxxxxx"
13    db_name = "mydb"
14    #Bucle infinito
15    while True:
16        #Probamos la conexion
17        try:
18            #Conectamos a MySQL
19            conn = pymysql.connect(host, user=name, passwd=password, db=db_name, connect_timeout=1)
20            connection = True
21            #Mientras haya conexion
22            while connection == True:
23                #Leemos el directorio (datos en local)
24                files = glob2.glob('/home/pi/Desktop/tfg/noconn_db/*')
25                #Delay minimo para edicion de ficheros leidos
26                time.sleep(0.1)
27                #Para cada fichero leido
28                for file in files:
29                    t = int(file[31:41])
30                    length = int(file[42])
31                    w = int(file[44:(44+length)])
32                    with conn.cursor() as cur:
33                        #Probamos la consulta SQL
34                        try:
```

```
35                              #Insertamos los datos en la base de datos SQL de MySQL
36                              cur.execute("insert into consumption (timestamp,w) values(%s,%s)",(t,w))
37                              conn.commit()
38
39                          except:
40                              #Si falla la consulta volvemos a intentar la conexion
41                              connection = False
42                      #Borramos el fichero
43                      os.remove(file)
44          except:
45              #Error de conexion
46                  print("connection error")
47
48  def main():
49      upload()
50
51  def destroy():
52
53  if __name__ == "__main__":
54      #Delay para iniciar el programa
55      time.sleep(1)
56      try:
57          main()
58      except KeyboardInterrupt:
59          destroy()
```

6.7.4 mqtt_upload.py

Función: Actualizar la sombra del objeto registrado en el IoT Core de Amazon Web Services vía MQTT, cuando haya conexión a Internet. Cada actualización se guarda en una base de datos DynamoDB directamente en la nube.

Código 6.11: Script de subida de datos de consumo por MQTT a IoT Core

```
1  import time
2  import os
3  import glob2
4  from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTShadowClient
5
6  #Funcion que se ejecuta al actualizar la sombra (vacía)
7  def myShadowUpdateCallback(payload, responseStatus, token):
8    return
9
10 #Funcion de subida continua de datos a la nube
11 def toIoTCore():
12     #Definicion de variables
13     SHADOW_CLIENT = "shadowClient"
14     HOST_NAME = "a3jr75faljmeqt-ats.iot.eu-west-1.amazonaws.com"
15     ROOT_CA = "AmazonRootCA1.pem"
16     PRIVATE_KEY = "38fe8c20c0-private.pem.key"
17     CERT_FILE = "38fe8c20c0-certificate.pem.crt"
18     SHADOW_HANDLER = "MyRPi"
19     #Configuracion del cliente
20     myShadowClient = AWSIoTMQTTShadowClient(SHADOW_CLIENT)
21     myShadowClient.configureEndpoint(HOST_NAME, 8883)
22     myShadowClient.configureCredentials(ROOT_CA, PRIVATE_KEY,CERT_FILE)
23     myShadowClient.configureConnectDisconnectTimeout(10)
24     myShadowClient.configureMQTTOperationTimeout(5)
25     myShadowClient.connect()
26     #Definicion del manejador de la sombra
27     myDeviceShadow = myShadowClient.createShadowHandlerWithName(SHADOW_HANDLER, True)
```

```
    #Bucle infinito
    while True:
        #Leemos los ficheros del directorio
        files = glob2.glob('/home/pi/Desktop/tfg/noconn_db/*')
        time.sleep(0.08)
        #Para cada fichero
        for file in files:
            t = int(file[31:41])
            length = int(file[42])
            w = int(file[44:(44+length)])
            #Creamos el JSON con los datos
            json = '{ "state": { "reported":{ "IoTObject": "RaspberryPi","Timestamp": '+str(t)+', "Data":{ "Power": '+ str(w) +', "Temperature": 25}}}}'
            #Actualizamos la sombra
            myDeviceShadow.shadowUpdate(json,myShadowUpdateCallback, 5)
            #Borramos el archivo
            os.remove(file)
            #Delay necesario: Si hay una caída en la red y subimos demasiados JSONs a IoT Core por minuto se produce un error
            time.sleep(0.1)

def main():
    toIoTCore()

def destroy():

if __name__ == "__main__":

    time.sleep(1)
    try:
        main()
    except KeyboardInterrupt:
        destroy()
```

7 Resultados

Se han obtenido unos resultados gráficos con respecto a la monitorización del consumo usando herramientas para visualizar los datos como Amazon Quicksight (Herramienta de AWS) o MatLab. Estas gráficas expuestas a continuación nos permiten conocer, principalmente:

1. Cómo varía el consumo de la vivienda en el tiempo
2. La potencia máxima histórica
3. La potencia media histórica
4. El consumo diario medio de la vivienda

El prototipo ha estado y está funcionando correctamente y se han cumplido los requisitos mínimos propuestos en este trabajo. Tenemos un sistema de gestión energética que mide los datos del consumo de la vivienda cada milisegundo y sube estos datos a la nube cada segundo (la media de cada segundo).

En la nube, estos datos de consumo, se envían a guardar en bases de datos de distinto tipo: SQL y NoSQL. Estos pueden ser consultados para su visualización, tratamiento, análisis y procesamiento posterior. Esto nos permite tomar medidas y acciones con el objetivo de mejorar la eficiencia energética de la vivienda.

Un ejemplo probado en este trabajo es la notificación al usuario mediante correo electrónico y SMS de una alerta de consumo excesivo que podría hacer superar la potencia contratada. La visualización del consumo en el ámbito doméstico permite, entre otras cosas:

1. Identificar en qué momentos del día se producen picos de consumo
2. Percibir las medidas de ahorro energético que se tomen en la vivienda
3. Concienciarse de los consumos excesivos en el ámbito doméstico
4. Tomar medidas efectivas para favorecer la eficiencia energética

7.1 Amazon Quicksight

Amazon Quicksight es la herramienta de Amazon que se ha usado para la monitorización de los datos de consumo. Esta herramienta nos permite hacer consultas a las bases de datos y visualizar los datos que deseemos.

7.1.1 Consumo histórico

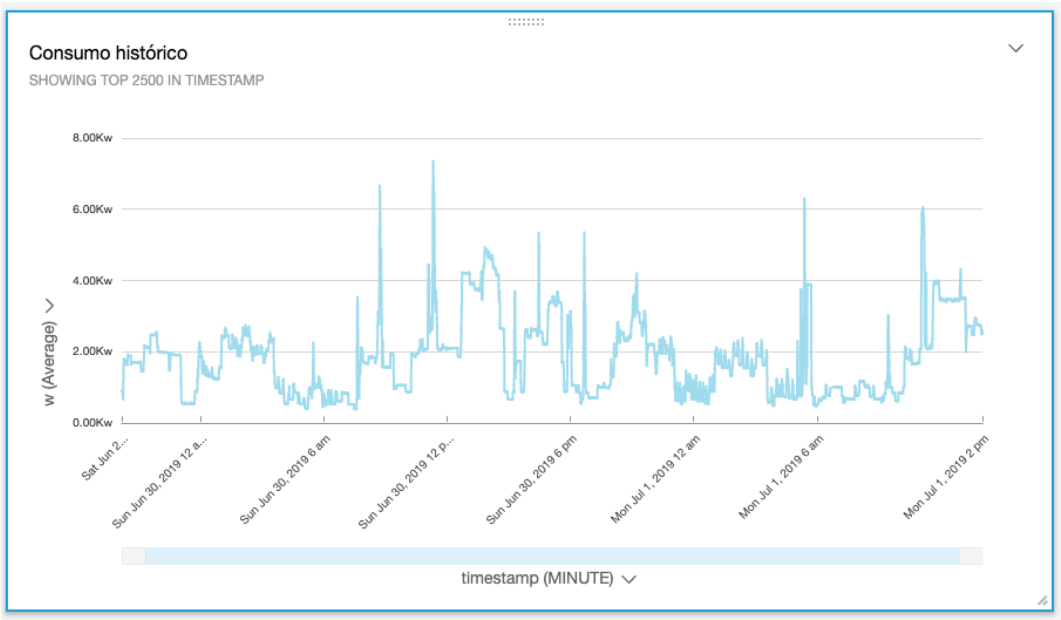


Figura 7.1: Consumo histórico por minuto (Últimos 2500 minutos).

7.1.2 Potencia máxima histórica

Figura 7.2: Potencia máxima histórica.

7.1.3 Potencia media histórica

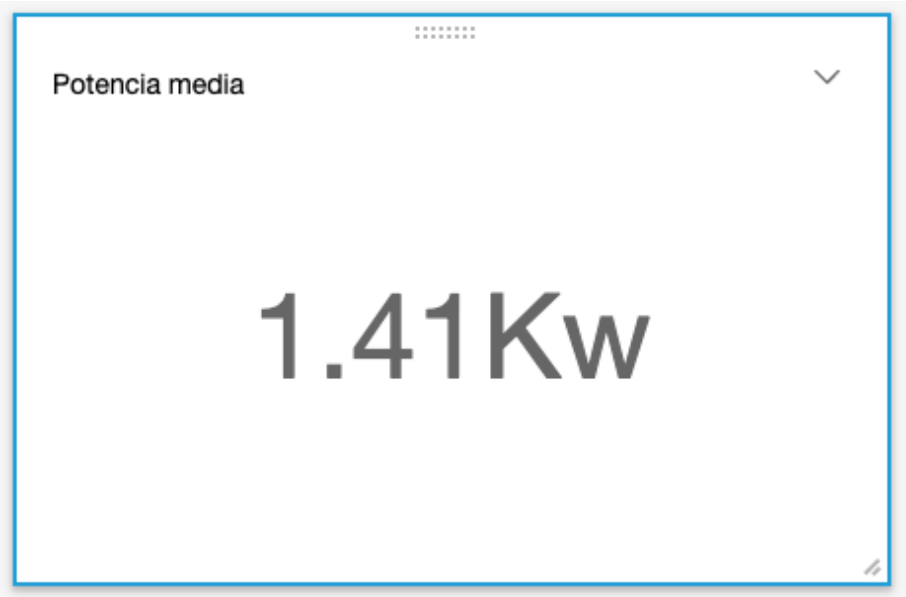


Figura 7.3: Potencia media histórica.

7.1.4 Consumo medio diario

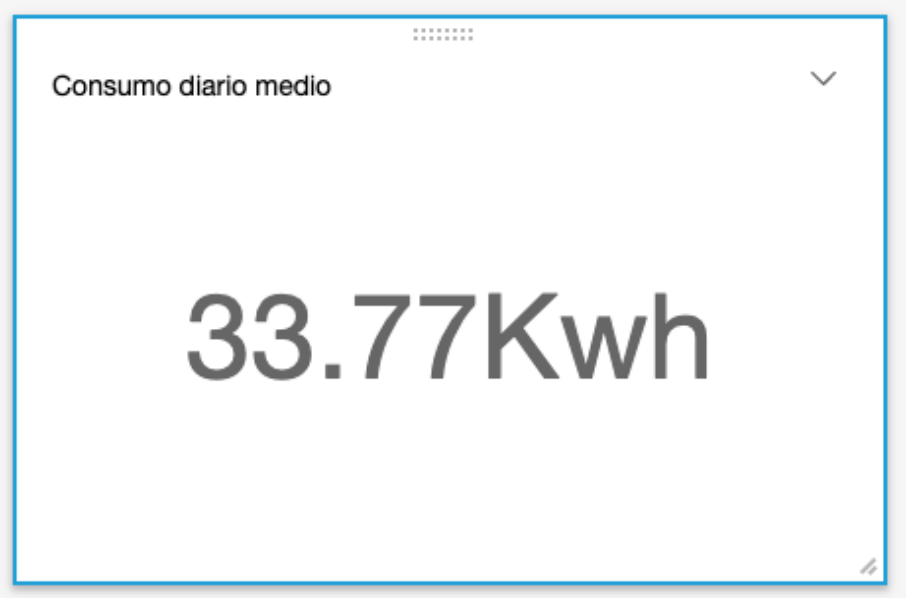


Figura 7.4: Consumo diario medio.

7.2 MatLab

De cara al tratamiento de datos en local se ha utilizado MatLab. Con esta herramienta podemos hacer consultas a bases de datos fácilmente. Para usar la aplicación para bases de datos de MatLab es necesario descargar un driver (ODBC/JDBC).

Una vez tenemos los datos en MatLab, podemos:

1. Visualizar los datos
2. Procesar los datos
3. Aplicar algoritmos de reconocimiento de patrones
4. Encontrar anomalías

Entre otras cosas. Un ejemplo de visualización del histórico del consumo por hora es el que se encuentra en la siguiente figura. Cada día tiene un color diferente. Podemos ver fechas en las que no se leyó el consumo porque el prototipo estaba realizando diferentes tareas para este trabajo.

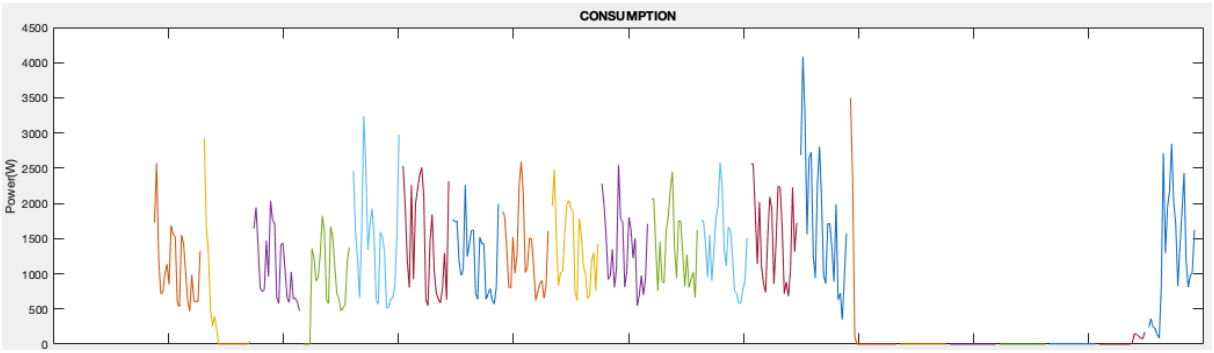


Figura 7.5: Histórico del consumo.

Gracias a estas posibilidades que nos ofrece MatLab, podríamos por ejemplo predecir el consumo de una vivienda siguiendo unos patrones aprendidos. Como podemos ver en la siguiente figura, son los consumos diarios de varios días superpuestos, existe una similitud y es muy probable que se pueda encontrar un patrón de consumo, así como patrones en determinados momentos del día.

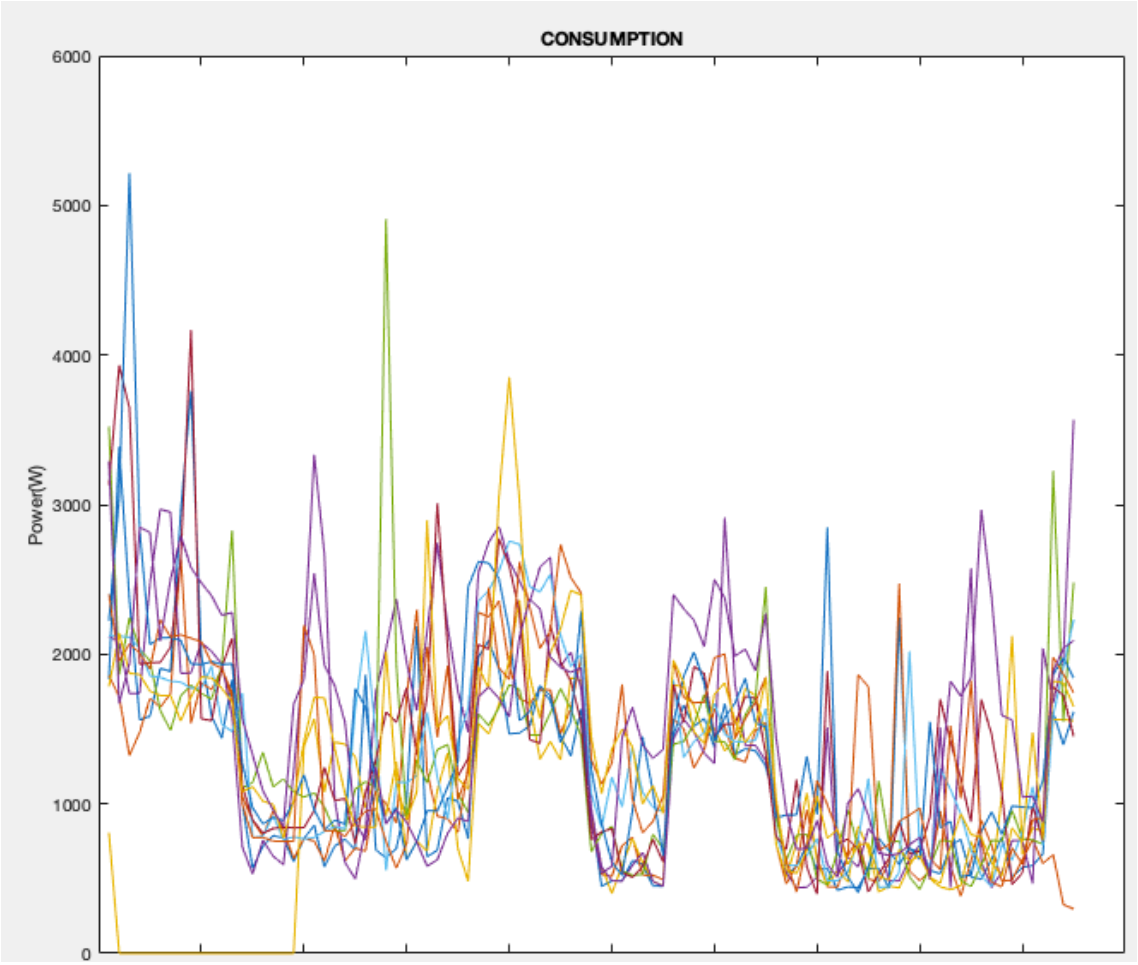


Figura 7.6: Consumos diarios superpuestos.

7.3 Notificaciones

Usando las acciones del IoT Core se ha conseguido recibir notificaciones SMS y en el correo electrónico. Esto se puede aplicar cuando haya un consumo excesivo que podría superar la potencia contratada y hacer que salten los plomos, por ejemplo.

Si se aplicaran algoritmos de aprendizaje se podrían notificar anomalías, encendidos de electrodomésticos y cualquier evento que ocurriese.

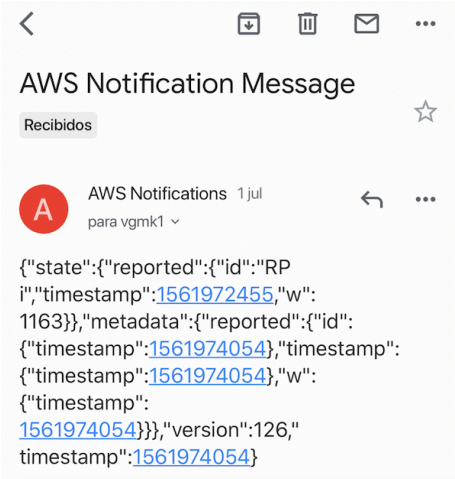


Figura 7.7: Notificación vía email de AWS.

8 Conclusiones

En primer lugar debemos destacar la importancia de las comunicaciones en cualquier ámbito hoy en día. En este trabajo hemos estudiado la gran cantidad de tecnologías y de protocolos de comunicación que existen y se ha comprobado que es fundamental la elección adecuada de estos aspectos a la hora de crear una aplicación útil. La tecnología 5G, sin lugar a dudas, va a marcar el futuro del Internet de las Cosas y, es probable, que cambie el modo de concebir el diseño y desarrollo de aplicaciones en Smart Home y en la Industria 4.0 (o 5.0).

Podemos decir que este trabajo ha servido para comprender la utilidad de los servicios en la nube y demostrar su funcionalidad. También, para entender y aprovechar las ventajas que nos ofrece el desarrollo de aplicaciones en la nube y cómo podemos mejorar la eficiencia energética mundial con las tecnologías de vanguardia.

Hemos comprendido que la eficiencia energética es un aspecto de gran importancia y está generando el auge del sector energético con el uso de energías renovables y con la mejora de los sistemas, que serán cada vez más eficientes gracias a proyectos similares a este.

En cuanto al prototipo, se ha realizado una instalación real de un sistema con el uso de sensores midiendo datos de consumo energético real. Se ha diseñado y programado un algoritmo que sube los datos cada segundo sin ningún tipo de pérdida a la nube, utilizando bases de datos de tipo SQL y NoSQL. Se han conseguido programar acciones en función de estos datos gracias a los servicios de Amazon Web Services. Se han visualizado estos datos tanto en la plataforma de AWS como en local. Este prototipo sirve de base para trabajos futuros de gestión energética en la plataforma AWS e, incluso, para otras plataformas.

Los resultados obtenidos sirven para visualizar estos datos, conocer el consumo energético de la vivienda y entender el potencial de desarrollo de aplicaciones en este sector. Con estos datos se pueden aplicar algoritmos avanzados de Machine Learning para predecir el consumo de la vivienda ya que podemos ver que existen patrones que pueden aprenderse.

Finalmente, comentaremos la proyección de este proyecto en los trabajos futuros que pueden desarrollarse con la base implementada.

8.1 Trabajos futuros

En este trabajo se ha planteado la posibilidad de incluir en el diseño del prototipo dos aspectos importantes en la gestión energética, como son:

1. Algoritmos de gestión energética

2. Gestión de la producción de energía renovable

8.1.1 Algoritmos de gestión energética (Machine Learning)

Una característica que aún está por desarrollar en el prototipo es la gestión en cuanto a actuar sobre dispositivos y producir medidas reales de ahorro energético. Este trabajo ha llegado hasta la monitorización y tratamiento básico de datos, lo que indirectamente puede generar una mejora en la eficiencia energética.

La plataforma de AWS ofrece herramientas suficientes para desarrollar sistemas en la nube que gestionen una vivienda inteligente, energéticamente hablando, al menos. Esta, junto con un conjunto de sensores y actuadores pueden mejorar la eficiencia energética de una vivienda directamente.

En la siguiente figura vemos el enchufe inteligente de Amazon, que podría ser un ejemplo de lo que estamos comentando y podría funcionar como actuador a la hora de enchufar y desenchufar dispositivos o electrodomésticos.

Figura 8.1: Enchufe inteligente de Amazon.

Con los resultados obtenidos tenemos una base para trabajar y aplicar algoritmos de análisis, predicción, aprendizaje, clasificación, etc. Hoy en día, las herramientas de Machine Learning son muy potentes y ofrecen un servicio efectivo. Estas herramientas pueden utilizarse en local o en la nube.

Hemos visto que descargábamos la base de datos en MatLab, lo que supone que podamos utilizar las herramientas de ML que esta aplicación nos ofrece. Sucede lo mismo con Python,

este lenguaje también dispone de librerías de ML.

Sin embargo, una de las claves de este trabajo es el procesamiento de estos datos en la nube. Amazon Web Services dispone de herramientas de Machine Learning y Deep Learning en su plataforma Amazon Sagemaker. Esta plataforma permite desarrollar modelos de aprendizaje automático de manera sencilla. Ejemplos de uso de esta herramienta se pueden encontrar en la web `https://docs.aws.amazon.com/sagemaker/latest/dg/howitworks-nbexamples.html`.

8.1.2 Gestión de la producción de energía renovable

En el diseño del prototipo se planteó la posibilidad de desarrollar un sistema que combinara la monitorización del gasto energético y de la producción energética. En el ámbito doméstico podría entrar en juego la producción mediante placas solares, haciendo el sistema aún más interesante desde el punto de vista medioambiental.

Aunque finalmente no se ha llegado a incluir esta funcionalidad, es un trabajo futuro muy interesante y que, sin duda, habría que investigar si se llevara a cabo un proyecto basado en el prototipo propuesto en este trabajo.

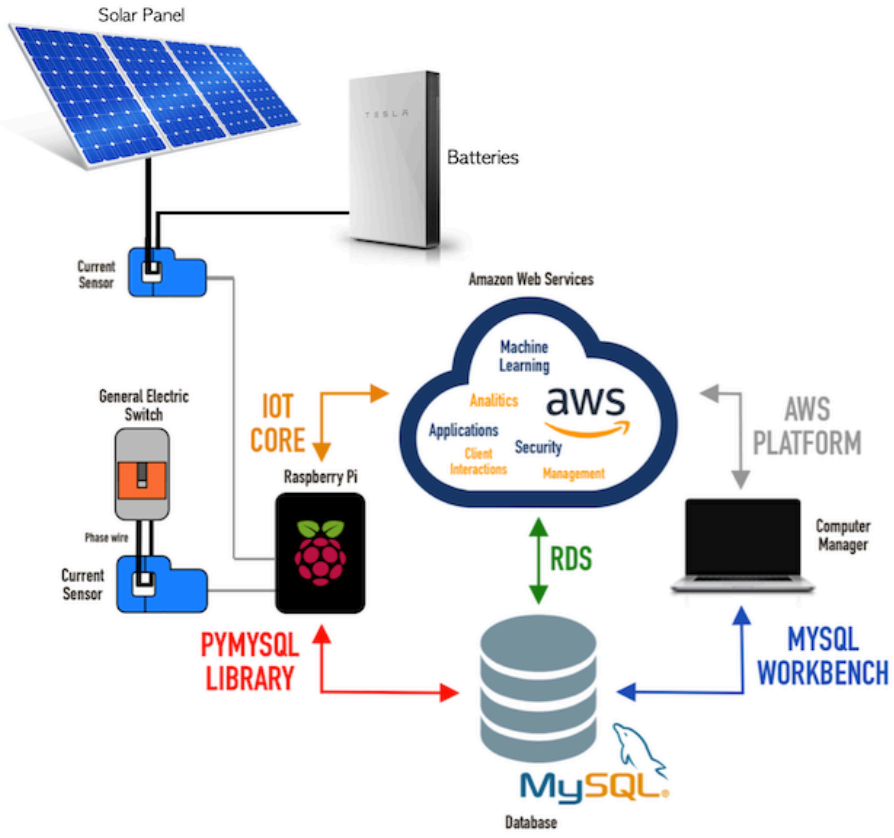


Figura 8.2: Estructura del diseño de un sistema de gestión del consumo y la producción energética renovable.

9 Asignaturas

Las siguientes asignaturas, especialmente, del Grado de Ingeniería Robótica cuentan con contenidos que han servido en gran medida para la realización de este trabajo:

1. *Comunicaciones*
2. *Sistemas empotrados*
3. *Procesadores integrados*
4. *Sensores e instrumentación*
5. *Tecnología eléctrica*
6. *Tecnología electrónica*
7. *Computadores*
8. *Algoritmia*
9. *Programación II*

Bibliografía

AlibabaCloud. (2018, Abril). *Iot communication protocol-compare-mqtt, dds, amqp, xmpp, jms, rest, coap.* Descargado 11/5/2019, de https://topic.alibabacloud.com/a/iot-communication-protocol-compare-mqtt-dds-amqp-xmpp-jms-rest-coap_3_79_31064989.html

AmazonWebServices. (2019, Mayo). *Módulo 1: Configuración de aws iot y envío de datos con el equipo de desarrollo.* Descargado 1/5/2019, de https://docs.aws.amazon.com/es_es/iot/latest/developerguide/iot-plant-module1.html

Bains, M. (2014, Junio). *5 things you must consider before you decide to adopt a nosql database such as mongodb.* Descargado 22/5/2019, de https://www.netsolutions.com/insights/5-things-you-must-consider-adopt-nosql-databases-mongodb/

EFOR. (2018, Mayo). *Tecnologías de comunicaciones para iot.* Descargado 15/4/2019, de https://www.efor.es/sites/default/files/tecnologias-de-comunicacioon-para-iot.pdf

F, I. (2019, Marzo). *Public cloud service comparison.* Descargado 12/5/2019, de http://comparecloud.in/?_lrsc=c276bb2a-7ec4-4a23-8f51-9ac3645823b8

Fernández, J. (2018). *Implantación de una solución de escritorios virtuales con openstack.*

Gemalto. (2019, Mayo). *Red 5g – características y usos de esta tecnología.* Descargado 29/5/2019, de https://www.gemalto.com/latam/telecom/inspiracion/5g

Knapp, E., y Samani, R. (2013). *Applied cyber security and the smart grid: Implementing security controls into the modern power infrastructure.* Syngress.

MSV, J. (2016, Mayo). *10 diy development boards for iot prototyping.* Descargado 24/4/2019, de https://thenewstack.io/10-diy-development-boards-iot-prototyping/

Sena, M. (2017, Marzo). *Cómo pasar de sql a nosql sin sufrir.* Descargado 30/5/2019, de https://medium.com/techwomenc/como-pasar-de-sql-a-nosql-sin-sufrir-e34dd22349e5

SunFounder. (2017, Noviembre). *Plus shield.* http://wiki.sunfounder.cc/index.php?title=Plus_Shield. Descargado 1/7/2019, de http://wiki.sunfounder.cc/index.php?title=Plus_Shield

Wikipedia. (2018a, Junio). *Consumo y recursos energéticos a nivel mundial.* Descargado 10/4/2019, de https://es.wikipedia.org/wiki/Consumo_y_recursos_energ%C3%A9ticos_a_nivel_mundial

Wikipedia. (2018b, Mayo). *Estrategia energética de la unión europea.* Descargado 15/4/2019, de https://es.wikipedia.org/wiki/Estrategia_energ%C3%A9tica_de_la_Uni%C3%B3n_Europea